துவக்க நிலையாளர்களுக்கான ஜாவா உரைநிரல் (JavaScript)

ச.குப்பன்

JS

துவக்க நிலையாளர்களுக்கான ஜாவாஉரைநிரல்

ச. குப்பன்

FreeTamilEbooks.com

துவக்க நிலையாளர்களுக்கான ஜாவாஉரைநிரல்

துவக்க நிலையாளர்களுக்கான ஜாவாஉரைநிரல்

துவக்க நிலையாளர்களுக்கான ஜாவாஉரைநிரல்

ச. குப்பன்

kuppansarkarai641@gmail.com

மின்னூல் வெளியீடு : FreeTamilEbooks.com

அட்டைப்படம் - லெனின் குருசாமி - gurulenin n@gmail.com

மின்னூலாக்கம் - ஐஸ்வர்யா லெனின் - aishushanmugam09@gmail.com

This book was produced using pandoc

பதிவிறக்கம் செய்ய - http://FreeTamilEbooks.com/ebooks/javascript_for_beginner

மின்னூல் வெளியீட்டாளர்: http://freetamilebooks.com

அட்டைப்படம்: லெனின் குருசாமி - guruleninn@gmail.com

மின்னூலாக்கம்: ஐஸ்வர்யா லெனின் - aishushanmugam09@gmail.com

மின்னூலாக்க செயற்திட்டம்: கணியம் அறக்கட்டளை - kaniyam.com/foundation

Ebook Publisher: http://freetamilebooks.com
Cover Image: Lenin Gurusamy - guruleninn@gmail.com
Ebook Creation: Iswarya Lenin - aishushanmugam09@gmail.com
Ebook Project: Kaniyam Foundation - kaniyam.com/foundation

This Book was produced using LaTeX + Pandoc

முன்னுரை

துவக்கநிலையாளர்களுக்கான ஜாவாஉரைநிரல் எனும் கணினி மொழிக்கான இந்தகையேடானது ஜாவாஉரைநிரல் எனும்கணினி மொழியின் 80% தலைப்பினை 20% நேரத்தில் கற்றுக்கொள்க எனும் ஒருவிதியைப் பின்பற்றுகிறது . இந்த அணுகுமுறையானது ஜாவாஉரைநிரல் எனும் கணினிமொழியின் மீது ஒரு முழுமையான கண்ணோட்டத்தை அளிக்கக்கூடும் என நான் நம்புகிறேன். இந்த கையேட்டில் ஜாவாஉரைநிரல் எனும் கணினிமொழி தொடர்பான சூரியனுக்குக் கீழே உள்ள அனைத்துவிவரங்களையும் கொண்டு வருவதற்காக முயற்சிக்கவில்லை. இந்த கையேடானது ஜாவாஉரைநிரல் எனும் கணினிமொழியின் முதன்மையான தகவல்களில் அதிக கவனம் செலுத்துகிறது, மேலும் சிக்கலான தலைப்புகளை எளிதாக்க முயற்சிக்கிறது. இந்த கையேட்டின் உள்ளடக்கங்கள் ஜாவாஉரைநிரல் எனும் கணினிமொழியின் அனைவரும் விரும்புகின்ற அடிப்படைகளைக் கற்றுக்கொள்ள உதவுக்கூடும் என நம்புகிறேன் முயற்சி வெற்றி பெற வாழ்த்துக்கள்

முனைவர் ச.குப்பன்

ஜாவாஉரைநிரல்(script) ஒருஅறிமுகம்

உலகின் மிகவும் பிரபலமான நிரலாக்க(கணினி) மொழிகளில் ஜாவா உரைநிரலும்**(script)** ஒன்றாகும்.துவக்கநிலையாளர்களுக்கான முதல் நிரலாக்க மொழியான இந்த ஜாவா உரைநிரல் ஆனது ஒரு சிறந்த கணினிமொழியாகும்.

பொதுவாக Node.jsஎன்பதைப் பயன்படுத்தி இணையதளங்கள், இணைய பயன்பாடுகள், சேவையாளர்களுக்கான பயன்பாடுகள் ஆகியவற்றினை உருவாக்க ஜாவாஸ்உரைநிரலைப் பயன்படுத்துகிறோம்.
ஆனால் ஜாவாஉரைநிரல் எனும் கணினி மொழியானது இந்த செயல்களுக்கு மட்டுமே உரியதென மட்டுப் படுத்தப்படவில்லை, அதனோடு மேலும் பல்வேறு செயல்களுக்கும் அதாவது React Native போன்ற கருவிகளைப் பயன்படுத்தி கைபேசி(CellPhone) பயன்பாடுகளை உருவாக்குதல், மீச்சிறு கட்டுபடுத்திகள்(Micro Controllers),பொருட்களுக்கான இணையம்(Internet of Things(IOT))ஆகியவற்றிற்கான நிரல்களை உருவாக்குதல்,

திறனுடைய கடிகார(Smart Watch) பயன்பாடுகளை உருவாக்குதல் என்பன போன்ற பல்வேறு செயல்களுக்கும் இதனை பயன்படுத்தி கொள்ளலாம் அடிப்படையில் இதனை கொண்டு இணையத்தில் எதையும் செய்ய முடியும் என்ற எல்லையில்லாத திறன் கொண்டு மிகவும் பிரபலமானதாக இருப்பதால், தற்போதைய சூழலில் இணையத்தில் புதியதாக தோன்றுகின்ற அனைத்திலும் ஏதாவது ஒருவகையில் ஒருவித ஜாவாஉரைநிரல் ஒருங்கிணைப்பினை கண்டிப்பாக கொண்டிருக்கும் என்பதே உண்மையான களநிலவரமாகும்.

ஜாவாஉரைநிரல் எனும் ஒரு நிரலாக்க மொழியானது:

உயர் நிலைகணினிமொழி: இது தான் இயங்குகின்ற கணினியின் விவரங்களைப் புறக்கணிக்க அனுமதிக்கின்ற சுருக்கமான விவரக் குறிப்புகளை வழங்குகிறது. இது கணினியின் ஒரு குப்பைகளை சேகரிப்பாளருடன் இணைந்து நினைவகத்தை தானாகவே நிர்வகிக்கிறது, எனவே C போன்ற பிற கணினி மொழிகளில் நினைவகத்தை நிர்வகிப் பதற்கு அதிக கவனத்தை வேண்டியுள்ளது என்பதற்கு பதிலாக இதில் குறிமுறைவரிகளின் மேம்பாடுகளில் மட்டுமே நம்முடைய கவனத்தை செலுத்தினால் போதும், மேலும் இதுஅதிக சக்தி வாய்ந்த மாறிகள், பொருட்களை(objects)சமாளிக்க அனுமதிக்கின்ற பல கட்டமைப்புகள் ஆகியவற்றினை வழங்குகிறது.

இயக்கநேர கணினிமொழி: ஒரு நிலையான மொழியானது

குறிமுறைவரிகளை இயந்திரமொழிக்கு மொழியாக்கம் செய்து தொகுக்கின்ற நேரத்தில் நாம் வேறு பல பயனுள்ள செயல்களை செய்துகொள்ளுமாறும் அவ்வாறு குறிமுறைவரிகளை இயந்திரமொழிக்கு மொழியாக்கம் செய்கின்ற பணியை இது இயக்க நேரத்திலேயே செயல்படுத்திகொள்ளுமாறும் இது கட்டமைக்கப்பட்டுள்ளது. இதிலும் பல்வேறு நன்மைகளும் உள்ளன ,தீமைகளும் உள்ளன, ஆயினும் இது இயக்கநேரவகை, தாமதமான ஒட்டுதல், பிரிதிபலித்தல், செயலி களின் நிரலாக்கம், பொருட்களுக்கான இயக்க நேர மாற்றம், நிறுத்தம் **மூடுதல்** போன்ற பல சக்திவாய்ந்த வசதிகளை வழங்குகிறது. அந்த செய்திகள் நமக்குத் தெரியாமல் இருந்தாலும் கவலைப்பட வேண்டாம் . இந்த கணினிமொழியை யாரும் எளிதாக நன்கு கற்றுகொள்ள முடியும்

இயக்கநேர மாறுகின்ற வகை: ஒரு மாறி தனியாக ஒரு வகையைச் செயல்படுத்தாது. இதில்எந்த ஒரு வகையையும் ஏதேனுமொரு மாறிக்கு மறுஒதுக்கீடு செய்யலாம், எடுத்துக்காட்டாக, ஒரு சரத்தை வைத்திருக்கின்ற மாறிக்கு முழு எண்ணைகூட ஒதுக்கீடுசெய்திடலாம்.
தளர்வான வகை: வலுவான வகைக்கு மாறாக, தளர்வான (அல்லது பலவீனமான) வகையிலான கணினி மொழிகள் ஒரு பொருளின் வகையைச் செயல்படுத்தாது, அதிக நெகிழ்வுத்தன்மையை அனுமதிக் கின்றன, ஆனால் நமக்கு வகைகளின் பாதுகாப்பினையும் வகைகளின் சரிபார்ப்பினையும் மறுக்கின்றன

(வகைஉரைநிரலானது(TypeScript) ஜாவா உரைநிரலின் மேல் கட்டமைப்பினை வழங்குகின்றது)

கணினிக்கான ஆணையை மாற்றதல்: இது பொதுவாக ஆணையை மாறுதல் செய்கின்ற ஒரு கணினி மொழி என்று அழைக்கப்படுகிறது. எடுத்துக்காட்டாக, C, Java அல்லது Go ஆகிய கணினி மொழிகளுக்கு மாறாக, ஒரு நிரல் இயங்குவதற்கு முன் அதற்கு ஒரு இயந்திர மொழியாக மாற்றிடுகின்ற நிலை இதில்தேவையில்லை. நடைமுறையில், செயல்திறன்போன்ற காரணங்களுக்காக இணைய உலாவிகள் ஜாவா உரைநிரலை இயக்குவதற்கு முன் இயந்திர மொழிக்கு மாற்றிடுகின்றன, ஆனால் வெளிப்படையானதாக இருப்பதால் கூடுதல் படிமுறை எதுவும் இதில்இல்லை

பல முன்னுதாரணம்: இந்த கணினிமொழியானது குறிப்பிட்ட எந்தவொரு நிரலாக்க முன்னுதாரணத்தையும் செயல்படுத்தாது, எடுத்துக்காட்டாக ஜாவாவைப் போன்றில்லாமல், இது பொருள் சார்ந்த நிரலாக்கத்தைப் பயன்படுத்து வதைத் தூண்டுகிறது அல்லது கட்டாய நிரலாக்கத்தை கட்டாயப்படுத்துகின்ற சி எனும் கணினிமொழியின் முன்மாதிரிகள் , புதிய (ES6 இன் படி) இனங்களின் இலக்கணம் ஆகியவற்றைப் பயன்படுத்தியும், பொருள் சார்ந்த முன்னுதாரணத்தைப் பயன்படுத்தியும் ஜாவாஉரைநிரலை எழுதலாம். ஜாவாஉரைநிரலை ஒரு செயலிகளின் நிரலாக்க பாணியில், அதன் முதல்-இன செயலிகளுடன் அல்லது ஒரு கட்டாய

பாணியில் (சி-எனும் கணினிமொழியைபோன்று) எழுதலாம். ஜாவாஎனும் கணினிமொழியைவிட ஜாவாஉரைநிரல் எனும் கணினி மொழியானது வேறு ஒன்றையையும் செய்யவில்லையே அதனால் ஜாவா எனும் கணினிமொழியே போதுமே என தவறாக இந்த கணினி மொழியை பற்றி எண்ணம் கொள்ளவேண்டாம் இது பெயருடன் மட்டுமே ஜாவா எனும் கணினி மொழியுடன் நெருங்கிய தொடர்புடையதே தவிர ஜாவாஉரைநிரல் எனும் கணினி மொழியை கொண்டு செய்யாத பணி இவ்வுலகில் எதுவுமே இல்லை வானமே இதன் எல்லை என்றவாறு நாம் இணையத்தில் செய்ய விரும்புகின்ற அனைத்து பணிகளுக்கும் இது பேருதவியாக விளங்குகின்றது அதனால் நாம் இந்த கணினிமொழியை மட்டுமே வைத்துகொண்டு நம்முடைய வாழ்க்கையை மிகச்சிறப்பாக ஆக்கிகொள்ளமுடியும்.

ஜாவாஉரைநிரலின்வரலாறு(History)

1995 இல் மிகச்சாதரணமான மிகவும் தாழ்ந்த நிலையில் தன்னுடைய பயனத்தினை துவக்கிய இந்த ஜாவாஉரைநிரல்எனும் கணினிமொழி யானது தற்போது மிக நீண்ட தூரம் பயனித்து அங்கிங்கு எனாதபடி இணையம் முழுவதும் எங்கெங்கும் பரவி எந்தவொரு இணைய பயன்பாடுகளிலும் இந்த கணினிமொழி இல்லையெனில் அவ்வாறான இணைய பயன்பாடே இல்லையென்றும் இணையமே இல்லையென்றும் ஆகிய மிகச்சிறந்த நிலைக்கு மிகஉயர்ந்து வளர்ந்துள்ளது.

இணைய உலாவிகளால் பூர்வீகமாக ஆதரிக்கப்பட்ட முதல் உரைநிரல் மொழி இதுமட்டுமேயாகும், இது வேறு எந்த கணினிமொழியையும் விட கூடுதலான நன்மையை கொண்டுள்ளதற்கான போட்டிகளில் வெற்றி பெற்றுள்ளது, இன்று நம்அனைவராலும் இணையபயன்பாடுகளை உருவாக்க நாம் பயன்படுத்தக்கூடிய ஒரே உரைநிரல் மொழி இந்த கணினிமொழி மட்டுமேயாகும்.

ஏராளமான அளவில் பிற கணினிமொழிகள் உள்ளன, ஆனால் அனைத்தும் இணைய பயன்பாடுகளை உருவாக்கவேண்டுமெனில் அவைகளை ஜாவாஉரைநிரலிற்கு மொழிமாற்றம்செய்திட வேண்டும் - அல்லது சமீபத்திய WebAssembly என்பதைகூட இதில் மொழிமாற்றம் செய்திடவேண்டும், ஆனால் இது மற்றொரு கதையாகும் இப்போதைக்கு தேவையில்லை.

தொடக்கத்தில், ஜாவாஉரைநிரல் எனும் கணினிமொழியானது இன்று இருப்பதை போன்று மிக சக்திவாய்ந்ததாக இல்லை, அதாவது இது ஆடம்பரமான அசைவூட்டங்கள், இயக்கநேர HTML என அறியப்பட்ட அற்புதங்கள் ஆகியவற்றிற்குகு மட்டுமே பயன்படுத்திகொள்ளுமாறு மட்டுப் படுத்தப் பட்டிருந்தது. ஆனால்தற்போதுஇணையதளம் கோரும் (தொடர்ந்து தேவைப்படுகிற) வளர்ந்து வருகின்ற அனைத்த தேவைகளுடன், உலகின் மிகவும் பரவலாகப் பயன்படுத்தப்படுகின்ற சுற்றுச்சூழல் அமைப்புகளில் ஒன்றின் தேவைகளுக்கு இடமளிக்கின்ற பொறுப்பையும் ஜாவாஉரைநிரல் எனும் கணினிமொழி கொண்டுள்ளது.

ஜாவாஉரைநிரல் தற்போது இணைய உலாவிக்கு வெளியேயும் பரவலாகப் பயன்படுத்திகொள்ளப்படுகிறது. அதாவது கடந்த சில ஆண்டுகளில் ஜாவா, ரூபி, பைதான், PHP ,போன்றபாரம்பரிய கணினி மொழிகளின் சேவையாளர்களின் களமான பின்புல

மேம்பாட்டிற்காக மட்டுமேயாக இருந்த இதனுடைய வாயில் தற்போது Node.js இன் எழுச்சியினை தொடர்ந்து பல்வேறு பயன்பாட்டிற்காகவும் பரந்து விரிந்து திறந்து கொண்டுவிட்டது. ஜாவாஉரைநிரல் ஆனது தற்போது கணினிமொழியின் ஆற்றல்மிக்க தரவுத்தளங்கள் போன்ற பல பயன்பாடுகளுக்கு அடிப்படையாக விளங்குகின்றது , மேலும் உட்பொதிக்கப்பட்ட பயன்பாடுகள், கைபேசி பயன்பாடுகள், தொலைகாட்சிபெட்டிகளின் பயன்பாடுகள்,போன்ற பலவற்றை உருவாக்குவது கூட இதன்மூலம் சாத்தியமாகின்ற நிலைக்கு இதுதற்போது முன்னேறியுள்ளது. இணைய உலாவியின் உள்பகுதியில் ஒரு சிறிய கணினி மொழியாகத் தொடங்கிய இது தற்போது வானமே இதனுடைய எல்லையென்றவாறு உலகின் மிகவும் பிரபலமான மொழியாக திகழ்கின்றது.

அனைவருக்கும் ஏற்ற ஜாவாஉரைநிரல்

சில நேரங்களில் ஜாவாஉரைநிரலினை பயன்படுத்தப்படும் சூழலில் அதனுடைய வசதிகளிலிருந்து இதனை பிரிப்பது மிகவும் கடினமாகும்

எடுத்துக்காட்டாக, பல குறிமுறைவரிகளின் எடுத்துக்காட்டுகளில் நாம் காணக்கூடிய console.log() எனும் கட்டளையானது ஜாவாஉரைநிரலின் குறிமுறை வரி அன்று. ஆயினும் மாறாக, இது இணைய உலாவியில் நமக்கு வழங்கப்பட்ட APIகளின் பரந்த நூலகத்தின் ஒரு பகுதியாகும். அதே வழியில், சேவையாளரில், Node.js வழங்கிய APIகளில் இருந்து ஜாவாஉரைநிரல் எனும் கணினி மொழியின் வசதிகளைப் பிரிப்பது என்பது சில நேரங்களில் மிகவும்கடினமாக இருக்கும்.

குறிப்பிட்ட வசதி React அல்லது Vueமூலம் வழங்கப்படுகிறதா? அல்லது இது “வெற்று ஜாவாஉரைநிரல்” அல்லது “வெண்ணிலா ஜாவாஉரைநிரல்” என்று அடிக்கடி அழைக்கப்படுகிறதா?

இங்கு ஜாவாஉரைநிரல், மொழியின் அடிப்படை பற்றி

விவாதிப்போம்.

இந்த கணினிமொழியை கற்கின்ற செயல்முறையானது வெளியே உள்ள செய்திகளுடன் சிக்கலாக்காமல், வெளிப்புற சுற்றுச்சூழல் அமைப்புகளால் வழங்கப்படுகிறது

ஜாவாஉரைநிரல் எனும்கணினிமொழியின் இலக்கணம்(Syntax)

இந்த சிறிய அறிமுகத்தில் இந்த கணினிமொழியின் இலக்கணமான 1.வெள்ளை(காலி)இடம்(white space) ,2.எழுத்துவடிவ உணர்திறன்(case sensitivity) ,3.நிலையுருக்கள் (literals), 4.குறிப்பிகள்(identifiers),5.குறிப்புரைகள் (comments) ஆகிய ஐந்தினை பற்றி மட்டும் இப்போது காண்போம்.

1.வெள்ளை(காலி) இடம்(white space)

ஜாவாஉரைநிரலின் வெள்ளை(காலி) இடமானது அர்த்தமுள்ளதாக கருதவில்லை. காலிஇடம் வரிகளின் மாற்றங்களின் கோட்பாட்டில் இருந்தாலும், நாம்விரும்புகின்ற எந்த பாணியிலும் இதனை சேர்க்கலாம். நடைமுறையில், பெரும்பாலும் நன்கு வரையறுக்கப்பட்ட பாணியை வைத்திருப்போம், மக்கள் பொதுவாகப் பயன்படுத்துவதைக் கடைப் பிடித்திடுவோம், மேலும் இதை ஒரு linter அல்லது பாணி போன்ற கருவியைப் பயன்படுத்தி செயல்படுத்தலாம்.எடுத்துக் காட்டாக, எப்போதும் இரு

எழுத்துக்களை உள்தள்ள விரும்புவதாக கொள்க அதற்காக இரு வெள்ளை(காலி) இடத்தினை மட்டும் விட்டுவிடுக.

2.எழுத்துவடிவ உணர்திறன்(case sensitivity)

ஜாவாஉரைநிரலானதுஎழுத்துஉணர்திறன் கொண்டது அதாவது ஒரு ஆங்கில எழுத்தினை அது சிறிய எழுத்தாஅல்லது பெரிய எழுத்தா என வெவ்வேறாக உணரும் திறன்கொண்டது.அதாவது ஆங்கிலத்தில் somethingஎன்பதும் Somethingஎன்பதும் வெவ்வேறானவை என உணர்ந்து கொள்ளும் திறன்கொண்டது .இந்த திறனையே இந்த மொழியானது குறிப்பிகளுக்கும் பயன்படுத்தி கொள்கின்றது

3.**நிலையுருக்கள்** (literals)

மூலக் குறிமுறைவரிகளில் எழுதப்பட்ட ஒரு மதிப்பை நிலையுருக்களாக நாம் வரையறுக்கலாம், எடுத்துக்காட்டாக, ஒரு எண், ஒரு சரம், ஒரு பூலியன் அல்லது இலக்கு நிலையுருக்கள் அல்லது கோவையின் நிலையுருக்கள் போன்ற மேம்பட்ட கட்டமைப்பை பின்வருமாறு எழுத்துப்பூர்வமாக வரையறுக்கலாம்.

5

‘Test’

true

[‘a’, ‘b’]

{color: ‘red’, shape: ‘Rectangle’}

குறிப்பிகள்(identifiers)

ஒரு குறிப்பி என்பது ஒரு மாறி, ஒரு செயலி, ஒரு பொருளினை(Object) அடையாளம் காண பயன்படுத்தக்கூடிய எழுத்துகளின் வரிசையாகும். இது ஒரு எழுத்துடன் ,$ என்றவாறு ஒரு டாலர் குறியீடுடன் அல்லது _ என்றவாறு ஒரு அடிக்கோட்டுடன் தொடங்கலாம் , மேலும் அது இலக்கங்களைக் கொண்டிருக்கலாம். ஒருங்குகுறியைப் பயன்படுத்தி, ஒரு கடிதத்தில்அனுமதிக்கப்படும் எழுத்துகளாககூட அதுஇருக்கலாம், எடுத்துக் காட்டாக, ஒரு ஈமோஜி

Test

test

TEST

_test

Test1

$test

மேலும் DOM கூறுகளைக் குறிப்பிட பொதுவாக டாலர் குறி பயன்படுத்தப் படுகிறது.சில பெயர்கள் ஜாவாஉரைநிரலின் உள்ளக பயன் பாட்டிற்காக ஒதுக்கப்பட்டுள்ளன, மேலும் அவற்றை குறிப்பிகளாகப் பயன்படுத்த முடியாது.

குறிப்புரைகள்(comments)

எந்தவொரு நிரலாக்கத்திலும் குறிப்புரைகள் மிக முக்கியமான

பகுதியாகும். அவைஎந்த நிரலாக்க மொழியிலும். முக்கியமானவை, ஏனெனில் அவை குறிப்பிட்டதொரு குறிமுறைவரிகளை காணும் மற்ற வர்களுக்கு (அல்லது நாமே) அந்த குறிமுறைவரிகளைப் படிக்கும் போது அதனை பற்றிய விளக்கம் பெறுவதற்கு பேருதவியாக திகழ்கின்றது குறிமுறைவரிகளுக்கு இடையில் அந்த குறிப்பிட்ட வரி குறித்து முக்கியத் தகவலைச் சேர்க்க இவ்வாறான சிறு குறிப்புரைகள் அனுமதிக்கின்றன. ஜாவாஉரைநிரலில், இதற்காக // எனும் குறி யீட்டினை ஒற்றை வரியிலான குறிப்புரையை எழுத பயன்படுத்தி கொள்ளலாம். // எனும் குறியீட்டிற்குப் பின் வரும் அனைத்தும் ஜாவாஉரைநிரல் மொழிபெயர்ப்பாளரால் குறிமுறை வரியாகக் கருதப்படு வதில்லை.பின்வருவதை போன்று இருக்கலாம்:

// இது ஒரு கருத்துரையாகும்

true //இது மற்றொரு கருத்துரையாகும்

மற்றொரு வகையில் கருத்துரைகள் பல வரிகளாக அமைந்திடும்போது /* என்ற குறியீட்டுடன் துவங்கி */ என்ற குறியீட்டுடன் முடிவடையுமாறு செய்திடலாம் . இவ்விரண்டு குறியீடுகளுக்கிடையில் உள்ள அனைத்தும் குறிமுறைவரிகளாகக் கருதப்படுவதிவில்லை:

/* இவை

சிலவகை

கருத்தரைகளாகும்

*/

அரைப்புள்ளிகள்(Semicolons)

ஜாவாஉரைநிரலில் உள்ள குறிமுறைவரி ஒவ்வொன்றின் செயலையும் இந்த அரைப்புள்ளி எனும்வாய்ப்பினை பயன்படுத்தி நிறுத்திடுமாறு செய்திடலாம்.ஜாவாஉரைநிரலின் மொழிபெயர்ப்பாளரானது நமக்காக அரைப்புள்ளிகளை அறிமுகப்படுத்துகின்ற அளவிற்கு மிக புத்திசாலியாக இருப்பதால் இதனை ஒரு சிறந்த வாய்ப்பாக பயன்படுத்தி கொள்க.
ஆயினும் பெரும்பாலான சந்தர்ப்பங்களில், நிரல்களில் அரைப்புள்ளி எனும் வாய்ப்பினை முழுவதுமாகத் தவிர்க்கலாம்.இந்த உண்மை மிகவும் சர்ச்சைக்குரியது, மேலும் நாம்பயன்படுத்திகொள்கின்ற குறிமுறை வரிகளில் எப்போதும் இந்த அரைப்புள்ளிகளை காணலாம் அல்லது காணாமலும் இருக்கலாம். கண்டிப்பாகத் தேவையில்லாதவரை எப்போதும் அரைப்புள்ளிகளைத் தவிர்ப்பதே சிறந்தது என பரிந்துரைக்கப் படுகின்றது.

மதிப்புகள்(Values)

hello எனும் சரமானது ஒரு மதிப்பாகும். 12 போன்ற எண்ணும் ஒரு மதிப்பாகும்.

hello, 12 ஆகிய மதிப்புகள். சரம் , எண் ஆகியவற்றின் வகைகளின் மதிப்புகளாகும்.

வகை(**type)** என்பது அதன் வகைகளின்(categories) ஒரு வகையான மதிப்பாகும், . ஜாவாஉரைநிரலில் பல்வேறு வகைகள் உள்ளன, அவற்றைப் பற்றி பின்னர் விரிவாக விவாதிப்போம். ஒவ்வொரு வகைக்கும் அதன் சொந்த பண்பியல்புகள் உள்ளன.

ஒரு மதிப்பிற்கு மேற்குறிப்பு ஒன்று தேவைப்படும்போது, அதனை ஒரு மாறிக்கு ஒதுக்குகின்றோம். மாறிக்கு ஒரு பெயர் இருக்கலாம், மேலும் மதிப்பு என்பது ஒரு மாறியில் சேமிக்கப்பட்டதாகும், எனவே நாம் பின்னர் அந்த மதிப்பை மாறியினுடைய பெயரின் மூலம் அனுகலாம்.

மாறிகள்(variables)

ஒரு மாறி என்பது ஒரு குறிப்பிக்கு(identifier) ஒதுக்கப்பட்ட மதிப்பாகும், எனவே அதை நிரலில் குறிப்பிட்ட பின்னர் பயன்படுத்தலாம்.

ஜாவாஉரைநிரலானது தளர்வானவகையா இருப்பதே இதற்குக் காரணமாகும், இதுவே அடிக்கடி உணருகின்ற ஒரு கருத்தமை வாகும்.எந்தவொரு மாறியையும் பயன்படுத்துவதற்கு முன்பு அதனை முதலில் அறிவிக்க (declared) வேண்டும். அவ்வாறு மாறிகளை அறிவிக்க நம்மிடம் const ,let ஆகிய இரண்டு முக்கிய வழிமுறைகள் உள்ளன. முதலில் const எனும் வழிமுறையை எவ்வாறு பயன்படுத்தி கொள்ளலாம் எனக்காண்போம்:

```
const a = 0
```

இப்போது let எனும் இரண்டாவது வழிமுறையை எவ்வாறு பயன்படுத்தி கொள்ளலாம் எனக்காண்போம்:

```
let a = 0
```

இவ்விரண்டு வழிமுறைகளுக்கும்இடையே என்ன வேறுபாடுகள் உள்ளன?

const ஒரு மதிப்புக்கான நிலையான மேற்குறிப்பை வரையறுக்கிறது. இதன் பொருள் பொதுவாக மேற்குறிப்பை மாற்ற முடியாது. அதற்கு புதிய மதிப்பையும் மீண்டும் ஒதுக்க முடியாது.

letஎனும் வழிமுறையை பயன்படுத்தி அதே மாறிக்கு புதிய மதிப்பை மீண்டும் ஒதுக்கலாம். ஆனால் constஇல் இதைச் செய்ய முடியாது எடுத்துகாட்டாக:

```
const a = 0

a = 1
```

இந்த கட்டளை வரியை செயல்படுத்தினால் உடன் திரையில் TypeError: எனும் ஒரு பிழையை காணலாம் ஏனெனில் const aஎனும் நிலையான வகை மாறிக்கு மீண்டும் வேறொரு மதிப்பை ஒதுக்கீடு செய்யமுடியாது. ஆனால் அதற்கு மறுதலையாக, let a எனும் மாறும் வகை மாறியைப் பயன்படுத்தி புதிய மதிப்பை மீண்டும் ஒதுக்கலாம் எடுத்துகாட்டாக:

```
let a = 0

a = 1
```

ஜாவாஉரைநிரல் எனும் கணினிமொழிபோன்று C போன்ற வேறுசில கணினிமொழிகளில் const என்பது “நிலையானது” என்று

பொருள்படாது, . குறிப்பாக, மதிப்பை மாற்ற முடியாது என்று பொருளன்று. அதாவது மாறிக்கான மதிப்பை மீண்டும் வேறொரு மதிப்பாக ஒதுக்கிடு செய்ய முடியாது என்பதே பொருளாகும். மாறி ஒரு பொருளினை அல்லது கோவையை சுட்டிக்காட்டினால் (பொருட்களை , கோவைகளைப் பற்றி பின்னர் பார்ப்போம்) பொருளின் அல்லது கோவையின் உள்ளடக்கம் சுதந்திரமாக மாற்றிகொள்ளலாம்.

const என்பதில் மாறிகளை அறிவித்திடும்போது மட்டும் தொடக்கநிலை மதிப்பளித்திட(initialized) வேண்டும்:

```
const a = 0
```

ஆனால்let என்பதில்மதிப்புகளை பின்னர் தொடக்கமதிப்பளித்திடலாம்

```
let a
```

```
a = 0
```

ஒரே அறிக்கையில் ஒரே நேரத்தில் பல மாறிகளை அறிவிக்கலாம்

```
const a = 1, b = 2
```

```
let c = 1, d = 2
```

ஆனால் ஒரே மாறியை ஒன்றுக்கு மேற்பட்ட முறை மீண்டும் அறிவிக்க முடியாது:

let a = 1

let a = 2

மேலேகூறியவாறு அறிவித்தால் “நகல் அறிவிப்பு(duplicate declaration)” எனும் பிழையை திரையில் காணலாம்.
அதனால் எப்போதும் constஎன்பதைமட்டும் பயன்படுத்தி கொள்க என பரிந்துரைக்கப் படுகின்றது, குறிப்பிட்ட அந்த மாறிக்கு மதிப்பை மீண்டும்(மறு)ஒதுக்கீடு செய்ய வேண்டும் என தேவைப்படும்போது மட்டுமே letஎன்பதை பயன்படுத்தி கொள்க. ஏனெனில் நமது குறிமுறைவரிகள் எவ்வளவு குறைவான சக்தி கொண்டதோ, அவ்வளவு சிறந்தது. மதிப்பை மறுஒதுக்கீடு செய்ய முடியாது என்றுநமக்குத் தெரிந்தால், அது குறைவான பிழைகளுக்கான ஒரு ஆதாரமாகும்.

இதுவரை const, let ஆகியஇரண்டுடன் எவ்வாறு பணிபுரிவது என்று பார்த்துவந்தோம், இந்நிலையில் 2015 வரை var எவ்வாறு பயன்படுத்தப் பட்டுவந்தது என தெரிந்து கொண்டால் நல்லது , 2015 வரையிலும் ஜாவாஉரைநிரலில் ஒரு மாறியை அறிவிக்க ஒரே வழி var என்பது மட்டுமே யாகும். ஆனால் இன்று, ஒரு நவீன குறிவரிமுறைகள் அனைத்தும் பெரும்பாலும் const , let ஆகிய இரண்டையும் மாறிகளை அறிவிக்கப் பயன்படுத்தி கொள்கின்றன. இங்கு இவைகளுக்கிடையிலான சில அடிப்படை வேறுபாடுகள்

மட்டுமே விவரிக்கப்பட்டுள்ளன, ஆனால் துவக்கநிலையாளர்கள் எனில், அதற்குமேலும் உள்ள வேறுபாடுகளை பற்றி கவலைப்படவேண்டாம் ஆயினும்பொதுவாக எப்போதும் குறிவரிகளில் const ,let ஆகிய இரண்டையும் பயன்படுத்திகொள்க என பரிந்துரைக்கப் படுகின்றது.

வகைகள்(Types)

ஜாவா உரைநிரலில் உள்ள மாறிகள் எந்த வகையையும் இணைப்ப தில்லை. ஏனெனில் அவை எந்தவகைகளையும் சேராதவைகளாகும் ஒரு மாறிக்கு சில வகையுடன் மதிப்பை ஒதுக்கியவுடன், வேறு எந்த வகையின் மதிப்பையும் புரலராக செய்ய, எந்தச் சிக்கலும் இல்லாமல், மாறியை மீண்டும் ஒதுக்கீடு செய்திடலாம். இவ்வாறான வகைகள் ஜாவாஉரைநிரலில் 1.பழமையான வகைகள்(primitive Types), 2.பொருள் வகைகள்(Object Types) ஆகிய இரண்டு முக்கிய வகைகளாக உள்ளன.

1. பழமையான வகைகள்(primitive Types)

எண்கள், சரங்கள், பூலியன்கள், குறியீடுகள், மேலும் பூஜ்யங்கள்(null), வரையறுக்கப்படாதவை (undefined) எனும் இரு சிறப்பு வகைகள்ஆகியன பழமையான வகைகளாகும் .

2.பொருள் வகைகள்(Object Types)

ஒரு பழமையான வகை அல்லாத எந்த மதிப்பும் (ஒரு சரம், ஒரு எண், ஒரு பூலியன், பூஜ்யம் அல்லது வரையறுக்கப்படாதது) ஒரு பொருள் வகையாகும்.பொருள் வகைகளுக்கு பண்பியல்புகள் உள்ளன,

மேலும் அந்த பண்பியல்புகளில் செயல்படக்கூடிய வழிமுறைகளும் உள்ளன.

பொருட்களைப் பற்றி பின்னர் விவாதிப்போம்.

வெளிப்பாடுகள்(Expressions)

வெளிப்பாடு என்பது ஜாவாஉரைநிரலின் குறிமுறைவரிகளின் ஒற்றை அலகு ஆகும், இது ஜாவாஉரைநிரல் இயந்திரமானது மதிப்பிடுசெய்து ஒரு மதிப்பை திரும்ப வழங்க உதவுகின்றது. வெளிப்பாடுகள் சிக்கலான தன்மையில் வேறுபடலாம். இப்போது முதன்மை வெளிப்பாடுகள் எனப்படும் மிகவும் எளிமையானவற்றிலிருந்து தொடங்குவோம்:

2

0.02

‘something’

true

false

this //இந்த நடப்புநோக்கம்

undefined

i //இங்கு iஎன்பது ஒரு மாறிஅல்லது மாறிலியாகும்

எண்கணித வெளிப்பாடுகளானவை ஒரு மாறியையும் , ஒரு செயற் குறியையும் எடுத்துக் கொண்டு (செயற்குறிகளின் அடிப்படையில் செயல்பட்டு)முடிவாக ஒரு விடையினை எண்ணாக கிடைக்கச் செய்திடுகின்றது:

1 / 2

i++

i -= 2

i * 2

சரத்தின் வெளிப்பாடுகள் ஒரு சரத்தை விளைவிக்கும் வெளிப்பாடுகளாகும்:

‘A’ + ‘string’

தருக்க வெளிப்பாடுகள் தருக்க செயற்குறிகளைப்(Operators) பயன் படுத்திகொள்கின்றன , பூலியன் மதிப்பைத் தீர்வுசெய்கின்றன:

a && b

a || b

!a

மேலும் மேம்பட்ட வெளிப்பாடுகள் பொருட்களையும், செயலிகளையும் கோவைகளையும் உள்ளடக்கியது, அவற்றைப்பற்றி பின்னர் விவாதிப்போம்

செயற்குறிகள்(Operators)

செயற்குறிகளானவை இரண்டு எளிய வெளிப்பாடுகளைப் பெறவும், அவற்றை ஒன்றிணைத்து மிகவும் சிக்கலான வெளிப்பாட்டை உருவாக்கவும் அனுமதிக்கின்றன. செயற்குறிகளை அவை பணிபுரியும் செயலின் அடிப்படையில் வகைப்படுத்தலாம். சில செயற்குறிகள் ஒரு செயலேற்பியில் மட்டும் செயல்படுகின்றன. பெரும்பாலான செயற்குறிகள் இரு செயலேற்பி களுடன் செயல்படுகின்றன ஒரே ஒரு செயற்குறியானது மூன்று செயலேற்பிகளுடன் செயல்படுகின்றன.
செயற்குறிகளுக்கான இந்த முதல் அறிமுகத்தில், நமக்கு மிகவும் தெரிந்த செயற்குறிகளை முதலில் அறிமுகப்படுத்தி கொள்வோம்:

மாறிகளை பற்றி விவாதிக்கம் போது ஏற்கனவே = எனும்ஒதுக்கீடு செயற்குறி ஒன்று அறிமுகப்படுத்த பட்டுள்ளது.ஒரு மாறிக்கு மதிப்பை ஒதுக்கீடுசெய்வதற்காக = எனும் செயற்குறியை பயன்படுத்தி வருகின்றோம்

let b = 2

அடிப்படை கணிதத்திலிருந்து நமக்கு ஏற்கனவே தெரிந்த இரும செயற்குறிகளின் மற்றொரு தொகுப்பை இப்போது காண்போம்.

(+)எனும் கூட்டல் செயற்குறி

const three = 1 + 2

const four = three + 1

சரங்களைப் பயன்படுத்தினால், + எனும்செயற்குறி சரங்களின் இணைப்பாகவும் செயல்படுகிறது, எனவே கவனமாக இந்த செயற்குறியை பயன்படுத்திடுக:

const three = 1 + 2

three + 1 // 4 என இதற்குவிடைகிடைக்கின்றது

'three' + 1 // three1 என இதற்குவிடைகிடைக்கின்றது

(-)எனும் கழித்தல் செயற்குறி

const two = 4 – 2

(/)எனும் வகுத்தல் செயற்குறி

இந்த செயற்குறிக்கு முந்தைய எண்ணை பிந்தைய எண்ணால் வகுத்திடும்போது கிடைக்கும் ஈவினை இது வழங்குகிறது:

const result = 20 / 5 // இதற்கான விடை (result) === 4 ஆகும்

const result = 20 / 7 // இதற்கான விடை(result) === 2.857142857142857 ஆகும்

இந்நிலையில் எந்தவொரு எண்ணையும் பூஜ்ஜியத்தால் வகுத்தால், ஜாவாஉரைநிரலானது எந்தவொரு பிழையையும் திரையில் காண்பிக்காது, ஆனால் முடிவிலி(Infinity) எனும் மதிப்பை (அல்லது முடிவிலியின் மதிப்பு எதிர்மறையாக இருந்தால் (-) என்றும்) வழங்குகின்றது.

எனும் மீதம் த1 / 0 // இதற்கான விடை முடிவிலியாகும்(Infinity)

-1 / 0 //இதற்கான விடை எதிர்மறை முடிவிலியாகும் ((-)Infinity)

(%) ருகின்ற செயற்குறி

மீதம் தருகின்ற செயற்குறி ஆனது பல பயன்பாட்டு நிகழ்வுகளில் மிகவும் பயனுள்ள கணக்கீடாக விளங்குகின்றது

const result = 20 % 5 //இதற்கான விடை(result) === 0ஆகும்

const result = 20 % 7 //இதற்கான விடை(result)=== 6 ஆகும்

பூஜ்ஜியத்தாலான மீதிஎண் எப்போதும் NaN ஆகும், இது " ஒருஎண் அன்று (Not a Number)" என்று பொருள்படும் சுருக்கமான சிறப்பு மதிப்பாகும்:

1 % 0 //இதற்கான விடை(NaN)ஆகும்

-1 % 0 //இதற்கான விடை(NaN)ஆகும்

(*) எனும் பெருக்கல் செயற்குறி

இரண்டு எண்களை பெருக்குவதற்காக இந்த செயற்குறி பயன்படுகின்றது

1 * 2 //இதற்கான விடை2ஆகும்

-1 * 2 //இதற்கான விடை-2 ஆகும்

() எனும் அடுக்குமுறையாக்கல் செயற்குறி**

இது இயக்கப்படுகின்ற முதல்எண்ணை இரண்டாவது எண்ணின் அடுக்குளாக உயர்த்துதலுக்கு பயன்படுகின்றது

1 ** 2 //இதற்கான விடை1 ஆகும்

2 ** 1 //இதற்கான விடை2ஆகும்

2 ** 2 //இதற்கான விடை4ஆகும்

2 ** 8 //இதற்கான விடை256

8 ** 2 //இதற்கான விடை64 ஆகும்

முன்னுரிமை(Precedence)

ஒரே வரியில் பல செயற்குறிகளைக் கொண்ட ஒவ்வொரு சிக்கலான அறிக்கையும் முன்னுரிமைச் சிக்கல்களை அறிமுகப்படுத்துகின்றது. அதனை பின்வரும் எடுத்துகாட்டில் காணலாம்

```
let a = 1 * 2 + 5 / 2 % 2
```

இதனுடைய விடை 2.5, ஆனால் ஏன் இந்த விடை கிடைக்கின்றது? இந்த எடுத்துகாட்டில்உள்ள செயற்குறிகளுக்குள் எந்தெந்த செயற்குறிகள் முதலில் செயல்படவேண்டும் , அதுவரை எவையெவை காத்திருக்க வேண்டும்? என்ற குழப்பம் ஏற்படுகின்றது அல்லவா!

சில செயற்குறிகள் மற்றவற்றை விட அதிக முன்னுரிமையைக் கொண்டுள்ளன. அதனால் செயற்குறிகளுக்கான முன்னுரிமை விதிகள் பின்வருமாறு அட்டவணையில் பட்டியலிடப்பட்டுள்ளன

செயற்குறிகள்	விளக்கம்
* / % ம	ுதலாவதாக பெருக்கலும் வகுத்தலும்
+ - இ	ரண்டாவதாக கூட்டலும் கழித்தலும்
=	

இறுதியாக ஒதுக்கீடுகள்

ஒரே மட்டத்தில் (+ , -) ஒன்றுக்குமேற்பட்ட செயற்குறிகள் இருந்தால் அவைகள் காணப்படுகின்ற வரிசையில் இடமிருந்து வலமாக, செயல்படுத்தப்பட வேண்டும்.இந்த விதிகளைப் பின்பற்றி, மேலே உள்ள செயற்குறிகளை பின்வருமாறான படிமுறைகளின்படி எடுத்துக்காட்டினை தீர்வு செய்திடமுடியும்:

let a = 1 * 2 + 5 / 2 % 2

let a = 2 + 5 / 2 % 2

let a = 2 + 2.5 % 2

let a = 2 + 0.5

let a = 2.5

ஒப்பீடுகள் (Comparisons)

ஒதுக்கீடு,கணித செயற்குறிகள் ஆகியவற்றிற்குப் பிறகு, மூன்றாவதாக நிபந்தனை செயற்குறிகளை அறிமுகப்படுத்திடுவோம். இரண்டு எண்கள் அல்லது இரண்டு சரங்களை ஒப்பீடுசெய்வதற்காக, பின்வரும் செயற்குறிகளைப் பயன்படுத்தலாம். ஒப்பீட்டு செயற்குறிகள் எப்பொழுதும் பூலியனைத் தருகின்றன, அது true அல்லது false ஆகிய இரண்டில் ஒரு மதிப்பாகமட்டுமே இருக்கும் . அவை சமத்துவமின்மை ஒப்பீட்டு செயற்குறிகளாகும்:

< எனில் “முந்தையது பிந்தையதைவிட குறைவானது”

<= எனில் “முந்தையது பிந்தையதைவிடகுறைவானது,அல்லதுசமமானது”

எனில் “முந்தையது பிந்தையதைவிட பெரியது”

= எனில் “முந்தையது பிந்தையதைவிட பெரியது, அல்லது சமமானது”

எடுத்துக்காட்டாக:

```
let a = 2

a >= 1 // இதற்கு true என விடை கிடைக்கின்றது
```

இவைகளைத் தவிர, நான்கு சமத்துவ செயற்குறிகளும் உள்ளன. அவை இரண்டு மதிப்புகளை ஏற்றுக்கொண்டு, ஒரு பூலியனைத் திருப்பித் தருகின்றன:

=== இது சமமானதாவென சரிபார்க்கிறது

!== இது சமமில்லாதவைகளா என சரிபார்க்கிறது

ஜாவாஉரைநிரலில் == , != ஆகிய இரு ஒப்பீடு செய்வதற்கான செயற்குறிகள் உள்ளன என்பதை நினைவில் கொள்க, ஆனால் இவ்விரண்டை மட்டும் ஒப்பீடு செய்வதற்கான செயற்குறிகளாக பயன்படுத்தி கொள்ளுமாறு பரிந்துரைக்கபடுகின்றது.ஏனெனில் ===,!==ஆகிய இவ்விரண்டும் குறிமுறைவரிகளில் சில நுட்பமான சிக்கல்களை தடுக்க பயன்படுகின்றன.

நிபந்தனைகள்(Conditionals)

ஒப்பீட்டு செயற்குறிகளை அடுத்து, இப்போது நாம் நிபந்தனைகளைப் பற்றி விவாதிப்போம்.

if**எனும் கூற்று**

ஒரு வெளிப்பாட்டில் மதிப்பீட்டின் முடிவைப் பொறுத்து நிரலானது ஒரு வழியை அல்லது வேறு வழியை எடுக்க ஒரு if எனும் கூற்றினை பயன்படுத்தி கொள்கிறது. இது ஒரு எளிமையான எடுத்துகாட்டாகும், பின்வருவது எப்போதும் செயல்படுத்தப்படுகிறது.

```
if (true) {
//ஏதோவொன்றுசெயல்படுகின்றது
}
```

மாறாக, பின்வருவது ஒருபோதும் செயல்படுத்தப்படுவதில்லை

```
if (false) {
//எதுவும் செய்வதில்லை (? ஒருபோதும் இல்லை?)
}
```

நிபந்தனையானது நாம் அனுப்பும் வெளிப்பாட்டை உண்மை அல்லது தவறான மதிப்புக்காக சரிபார்க்கிறது. நாம்ஒரு எண்ணைக் கடந்து சென்றால், அது 0 ஆக இருக்கும் வரை எப்போதும் உண்மை என மதிப்பிடப்படும். நாம் ஒரு சரத்தை கடந்து சென்றால், அது வெற்று சரமாக இல்லாவிட்டால், அது எப்போதும் true என்பதாக மதிப்பிடப்படும். இவை பூலியனுக்கு இனமாற்ற வகைகளுக்கான பொதுவான விதிகளாகும். இதில்இருதலைஅடைப்புகுறிகள் இருப்பதைக் கவனித்திடுக. இது ஒரு தொகுப்பு என்று அழைக்கப்படுகிறது, மேலும் இது வெவ்வேறு கூறுகளின் பட்டியலைத் தொகுக்கப் பயன்படுகிறது.

ஒரு ஒற்றை கூற்றினை எங்கு வேண்டுமானாலும் ஒன்றாக தொகுத்து வைக்கலாம். நிபந்தனைகளுக்குப் பிறகு செயல்படுத்த ஒரு கூற்று நம்மிடம் இருந்தால்,தொகுப்பினைத் தவிர்த்துவிட்டு, அந்த கூற்றினை எழுதலாம் எடுத்துகாட்டாக:

```
if (true) doSomething()
```

ஆனால் எப்போதும் தெளிவாக இருக்க இருதலை அடைப்புகுறிகளைப் பயன்படுத்திகொள்ளுமாறு பரிந்துரைக்கப்படுகின்றது.

Else எனும் கூற்று

அடுத்து if எனும் கூற்றிற்கு இரண்டாவது பகுதியாக இந்த else

என்பதை காண்போம். if எனும் கூற்றானது false:என இருந்தால் செயல்படுத்தி டு மாறு if எனும் நிபந்தனை கூற்றுடன் இந்த else என்பதை இணைத்திடலாம்:

```
if (true) {
//ஏதோவொன்றுசெயல்படுகின்றது
} else {
//பொய்என இருந்தால் வேறு ஏதோவொன்றுசெயல்படுகின்றது
}
```

இந்நிலையில் ஒரு கட்டளைவரியில் else எனும் ஒரு கூற்றினை ஏற்றுக்கொண்டால் அதன்பிறகு, அதற்குள் இன்னொரு கூற்றினை சேர்க்கலாம்:

```
if (a === true) {
//ஏதோவொன்றுசெயல்படுகின்றது
} else if (b === true) {
//பொய்என இருந்தால் வேறு ஏதோவொன்றுசெயல்படுகின்றது
} else {
//முந்தையதற்கு திரும்பி செல்கின்றது
}
```

சரங்கள்(Strings)

சரம் என்பது தொடர்ச்சியான எழுத்துகளாகும். மேற்கோள்கள் அல்லது இரட்டை மேற்கோள்களுக்குள் இணைக்கப்பட்ட ஒரு சரத்தினை சொற்களாகவும் வரையறுக்கலாம்:

‘A string’

"Another string

பொதுவாக நிரலாளர்கள்தனிப்பட்ட முறையில் எல்லா நேரங்களிலும் ஒற்றை மேற்கோள்களை விரும்புகின்றனர், மேலும் பண்பியல்புகளை வரையறுக்க HTML இல் இரட்டை மேற்கோள்களை மட்டுமே பயன்படுத்திடுகின்றனர். பின்வருமாறு மாறிக்கு ஒரு சர மதிப்பை ஒதுக்கிடு செய்திடலாம்:

const name = ‘Kuppan’

ஒரு சரத்தின் length எனும் பண்பியல்புகளைப் பயன்படுத்தி அதன் நீளத்தை தீர்மானிக்கலாம்:

‘Kuppan’.length //இதற்கான விடை 6 ஆகும்

const name = ‘Kuppan’

name.length // இதற்கான விடை 6 ஆகும்

இது ஒரு வெற்று சரம்: '' . இதன் length எனும்பண்பியல்பு 0 ஆகும்:

''.length // இதற்கான விடை 0ஆகும்

- எனும் செயற்குறியைப் பயன்படுத்தி இரண்டு சரங்களை ஒன்றிணைக்கலாம்

"A" + "string"

மாறிகளை இடைக்கணிப்பு செய்திட + எனும்செயற்குறியைப் பயன்படுத்தலாம்:

const name = 'Kuppan'

"My name is" + name // என்னுடைய பெயர்My name is Kuppan

சரங்களை வரையறுக்க மற்றொரு வழி, சரியெனும்பிற்குறிக்குள்(backtick) வரையறுக்கப்பட்ட மாதிரிப்பலக மதிப்புருகளைப் பயன்படுத்துவதாகும். பலவரிகளின் சரங்களை மிகவும் எளிமையாக்க அவை மிகவும் பயனுள்ளதாக இருக்கின்றன. ஒற்றை அல்லது இரட்டை மேற்கோள்கள் மூலம் பல வரிகளின் சரத்தை எளிதில் வரையறுக்க முடியாது: விடுபடுகின்ற எழுத்துகளைப் பயன்படுத்த வேண்டும்.இந்த பிற்குறியின் மூலம் ஒரு மாதிரி பலகம் திறக்கப்பட்டதும், சிறப்பு எழுத்துகள் இல்லாமல், புதிய

வரியை உருவாக்க Enterஎனும் விசையை அழுத்திடுக, அது அப்படியே எழுதப்படும்

```
const string = `Hey
this
string
is awesome!`
```

மாதிரிபலக எழுத்துக்கள் மிகச் சிறந்தவை, ஏனெனில் அவை எளிதான வழியை வழங்குகின்றன. மாறிகளின் வெளிப்பாடுகளை சரங்களாக இடைக்கணிப்பு செய்திடலாம் .${…} எனும் தொடரியலைப் பயன்படுத்தி அவ்வாறு செய்திடலாம்:

```
const var = ‘test’
const string = something ${var}
// ஏதோவொன்றினை சரிபார்த்தல்(test)
```

$ { } என்றவாறான இருதலைஅடைப்பு குறிகளுக்குள் எதையும் சேர்க்கலாம், மிகமுக்கியமாக வெளிப்பாடுகளை கூட சேர்க்கலாம்

```
const string = something ${1 + 2 + 3}
const string2 = `something
${foo() ? ‘x’ : ‘y’}`
```

கோவைகள் (Arrays)

கோவை என்பது பல உறுப்புகளின் ஒருதொகுப்பாகும். ஜாவாஉரைநிரலில் உள்ள கோவைகள் அவற்றின் சொந்த வகை அல்ல. கோவைகள் பொருட்களாகும்(Object). பின்வருமாறான இரு வெவ்வேறு வழிகளில் வெற்று கோவையை நாம் துவக்கலாம்

```
const a = [ ]
const a = Array()
```

முதலாவதாக, கோவை நிலையுருவின் இலக்கணத்தின்படி பயன்படுத்தப் படுகிறது. இரண்டாவது கோவையானது முன்கூட்டியே உள்ளமைக்கப் பட்ட செயலியைப் பயன்படுத்துகிறது. பின்வரும் இலக்கணத்தின் மூலம் நாம்கோவையை முன்கூட்டியே நிரப்பலாம்

```
const a = [1, 2, 3]
const a = Array.of(1, 2, 3)
```

ஒரு கோவையில் எந்தவொரு மதிப்பையும் வைத்திருக்க முடியும், வெவ்வேறு வகைகளின் மதிப்பு கூட ஒரு கோவையில் வைத்திருக்க

முடியும்

const a = [1, ‘Kuppan o’, [‘a’, ‘b’]]

நாம் ஒரு கோவையை மற்றொரு கோவையுடன் சேர்க்க முடியும் என்பதால், பல பரிமாண கோவைகளை உருவாக்கலாம், அவை மிகவும் பயனுள்ள பயன்பாடுகளைக் கொண்டுள்ளன (எ.கா. அணி):

const matrix = [

[1, 2, 3]

[4, 5, 6],

[7, 8, 9]

]

matrix[0][0] //இது ஒரு பூஜ்ஜிய அணிக்கோவையாகும்

matrix[2][0] // இது ஒரு பூஜ்ஜியமில்லாத அணிக்கோவையாகும்

ஒரு கோவையியின் எந்தவொரு உறுப்பையும் அதனுடைய சுட்டுவரிசையைக் குறிப்பிடுவதன் மூலம் அணுகலாம், பின்வருது பூஜ்ஜியத்திலிருந்து தொடங்குகிறது:

a[0] // இது ஒரு பூஜ்ஜியகோவையாகும்

a[1] //இது ஒன்று என்ற எண்ணுள்ள கோவையாகும்

a[2] //இது இரண்டு என்ற எண்ணுள்ள கோவையாகும்

இந்த இலக்கணத்தின் மூலம் மதிப்புகளின் தொகுப்புடன் புதிய கோவையை துவக்கலாம், இது முதலில் 12 உறுப்புகளின் கோவையைத் துவக்குகிறது, மேலும் ஒவ்வொரு உறுப்பையும் 0 எனும்எண்ணுடன் நிரப்புகிறது:

Array(12).fill(0)

கோவையின் lengthஇன் பண்பியல்புகளைச் சரிபார்ப்பதன் மூலம் அதனுடைய உறுப்புகளின் எண்ணிக்கையைப் பெறலாம்:

const a = [1, 2, 3]

a.length // இந்த கோவையானதுமூன்று உறுப்புகளின் நீளம் கொண்டதாகும்

கோவையின் lengthஐ குறிப்பிட்டு அமைக்கலாம் என்பதை நினைவில் கொள்க. கோவையின் தற்போதைய திறனை விட பெரிய எண்ணை ஒதுக்கினால், எதுவும் நடக்காது. அதற்குபதிலாக ஒரு சிறிய எண்ணை ஒதுக்கினால், கோவையானது அந்த நிலையோடு துண்டாக்கப்படும்:

const a = [1, 2, 3]

a //[1, 2, 3]

a.length = 2

a //[1, 2]

கோவையில் ஒரு உருப்படியை எவ்வாறு சேர்ப்பது

வழக்கமாக push() எனும் வழிமுறையைப் பயன்படுத்தி கோவையின் முடிவில் ஒரு உறுப்பைச் சேர்க்கலாம்:

```
a.push(4)
```

ஆயினும் unshift() எனும் வழிமுறையைப் பயன்படுத்தி கோவையின் தொடக்கத்தில் ஒரு உறுப்பைச் சேர்க்கலாம்

```
a.unshift(0)
a.unshift(-2, -1)
```

கோவைலிருந்து ஒரு உருப்படியை எவ்வாறு அகற்றுவது

வழக்கமாக pop() எனும் வழிமுறையைப் பயன்படுத்தி கோவையின் முடிவில் இருந்து ஒரு உருப்படியை அகற்றலாம்:

```
a.pop()
```

ஆயினும் shift()எனும் வழிமுறையைப் பயன்படுத்தி ஒரு கோவையின் தொடக்கத்திலிருந்து ஒரு உருப்படியை அகற்றலாம்:

```
a.shift()
```

இரண்டு அல்லது அதற்கு மேற்பட்ட கோவைகளை எவ்வாறு ஒன்றிணைப்பது

வழக்கமாக concat() எனும் வழிமுறையைப் பயன்படுத்தி ஒன்றுக்கு மேற்பட்ட கோவைகளை ஒன்றிணைக்கலாம்:

```
const a = [1, 2]

const b = [3, 4]

const c = a.concat(b) //[1,2,3,4]

a //[1,2]

b //[3,4]
```

இதற்காக (…) எனும் விரித்திடும் செயற்குறியையும் இந்த வழிமுறையில் பயன்படுத்தலாம்:

```
const a = [1, 2]

const b = [3, 4]

const c = […a, …b]

c //[1,2,3,4]
```

கோவையில் ஒரு குறிப்பிட்ட உருப்படியை எவ்வாறு கண்டுபிடிப்பது

அதற்காக ஒருகோவையில் find() எனும்வழிமுறையைப் பயன்படுத்தலாம்

```
a.find((element, index, array) => {

// true அல்லது false ஆகிய இரண்டு செய்திகளில்ஒன்றினை வழங்கும்

})
```

நாம் தேடிடும் உருப்படியை கண்டுபிடித்திட முடிந்தால் உடன் trueஎன்ற செய்தியை வழங்குகின்றது. உறுப்பினை

கண்டுபிடிக்கப்படவில்லை எனில் false என்ற செய்தியை வழங்கும்..

பொதுவாகப் பயன்படுத்தப்படும் இலக்கணம் பின்வருமாறு:

```
a.find(x => x.id === my_id)
```

மேலே உள்ள கட்டளை வரியானது .id === my_id ஐக் கொண்ட கோவையில் முதல் உறுப்பை வழங்குகின்றது.

அதனோடு findIndex() எனும் வழிமுறையானது find() எனும் வழிமுறைக்கு ஒத்ததாக செயல்படுகிறது, ஆனால் trueஎன இருந்தால் முதல் உருப்படியின் சுட்டுவரிசையை வழங்குகிறது, மேலும் கண்டு பிடிக்கப் படவில்லை என்றால், அது வரையறுக்கப்படவில்லையென்பதால் falseஎனும் செய்தியை வழங்குகின்றது:

```
a.findIndex((element, index, array) => {

//true அல்லது false ஆகிய இரண்டு செய்திகளில்ஒன்றினை வழங்கும்

})
```

includes() என்பது இதே பணிக்கான மற்றொரு வழிமுறையாகும் :

```
a.includes(value)
```

ஒரு மதிப்பைக் கொண்டிருந்தால் trueஎனும் செய்தியை வழங்குகின்றது.

```
a.includes(value, i)
```

இதில் ஒரு கோவையில் உருப்படியை தேடிடும்போது i என்ற நிலைக்குப் பிறகு a என்பது மதிப்பைக் கொண்டிருந்தால் trueஎன வழங்குகின்றது.

மடக்கிகள்(Loops)

ஜாவாஉரைநிரலின் முக்கிய கட்டுப்பாட்டு கட்டமைப்புகளில் மடக்கிகளானவை முக்கியமான ஒன்றாகும்.ஒரு மடக்கி மூலம், குறிப்பிட்ட நிரலை நாம் எத்தனை முறை இயக்க வேண்டும் என்பதற்காக, ஒரு குறிமுறைவரிகளின் தொகுப்பினைகொண்டு குறிப்பிட்ட தானியக்கமாக்கி, காலவரையின்றி மீண்டும் மீண்டும் செயற்படுத்தலாம். ஜாவாஉரைநிரல் மடக்கிகள் மூலம் நமக்கு தேவையான பல்வேறு செயல்களை திரும்பதிரும்ப செய்வதற்கு பல்வேறு வழிகளை வழங்குகிறது.
இங்கு while எனும் மடக்கிகள் , for எனும் மடக்கிகள் ,for..of எனும் மடக்கிகள் ஆகிய மூன்றில் மட்டும் கவனம் செலுத்திடுவோம்

1.while மடக்கிகள்

while மடக்கி என்பது நிரலின் வாயிலாக குறிப்பிட்ட செயலை எளிதாக மடக்கி திரும்ப திரும்ப செயல்படச் செய்வதற்காக உதவுகின்ற ஜாவா உரைநிரலின் ஒரு கட்டமைப்பாகும். அதற்காக while எனும் திறவுச் சொல்லுக்குப் பிறகு நாம் ஒரு நிபந்தனையைச் சேர்த்திடலாம், மேலும் குறிப்பிட்ட நிபந்தனை உண்மையாக(true)

ஆகும் வரை இயங்கிடுமாறு ஒரு தொகுப்பினை வழங்கிடலாம். எடுத்துகாட்டாக:

```
const list = [‘a’, ‘b’, ‘c’]
let i = 0
while (i < list.length) {
console.log(list[i]) //மதிப்பு
console.log(i) //iசுட்டுவரிசை
i = i + 1
}
```

இந்நிலையில் break எனும் திறவுச்சொல்லைப் பயன்படுத்தி பின்வருமாறானwhile எனும் மடக்கிகளில் குறுக்கிடலாம்

```
while (true) {
if (somethingIsTrue) break
}
```

ஒரு சுழலின் நடுவில் தற்போதைய மறு செய்கையைத் தவிர்க்க வேண்டும் என்று முடிவு செய்தால்,

continue எனும் கட்டளையைப் பயன்படுத்தி அடுத்த மறு செய்கைக்கு தாவிச் சென்றிடுமாறு செய்திடலாம்:

```
while (true) {
if (somethingIsTrue) continue
//வேறு ஏதாவது செய்க
}
```

அதேwhileஎனும் மடக்கியை பயன்படுத்தி செய்ததைபோன்று , do..whileஎனும் மடக்கிகளை பயன்படுத்தியும் செயற்படுத்தலாம். இந்த மடக்கிகளில் தொகுப்பான குறிமுறைவரிகள் செயல்படுத்தப்பட்ட பிறகு நிபந்தனை மதிப்பீடு செய்யப்படுவதைத் தவிர, இது அடிப்படையில் whileஎனும் மடக்கி செய்தை போன்றே செயல்படுகின்றது.இதன் பொருள் இந்த மடக்கிகளில் குறிமுறைவரிகளின் தொகுப்பு எப்போதும் ஒரு முறையாவது செயல்படுத்தப்படும் என்பதேயாகும்.எடுத்துகாட்டாக

```
const list = [‘a’, ‘b’, ‘c’]
let i = 0
do {
console.log(list[i]) //மதிப்பு
console.log(i) //சுட்டுவரிசை
i = i + 1
} while (i < list.length)
```

2.for எனும் மடக்கிகள்

ஜாவாஉரைநிரலில் இரண்டாவது மிக முக்கியமான மடக்கிகளின் அமைப்பு for எனும் மடக்கிகள் ஆகும். இதனை பயன்படுத்தி கொள்வதற்காக வெனfor எனும் திறவுச்சொல்லைப் பயன்படுத்தி கொள்ளலாம், இந்த மடக்கிகளைசெயல்படுத்திடுவதற்காக துவக்க நிலை(initialization), நிபந்தனை(condition), பகுதி அதிகரித்தல்(increment part)ஆகிய மூன்று வழிமுறைகளின் ஒரு தொகுப்பை கடத்தலாம்: எடுத்துகாட்டாக:

```
const list = [‘a’, ‘b’, ‘c’]
for (let i = 0; i < list.length; i++) {
console.log(list[i]) //மதிப்பு
console.log(i) //சுட்டுவரிசை
}
```

இதில் whileஎனும் மடக்கிகளைப் போன்றே, breakஎனும் கட்டளையைப் பயன்படுத்தி இதனுடைய செயலின் இடையில் குறுக்கிடலாம் மேலும் continue எனும் கட்டளையைப் பயன்படுத்தி ஒரு forஎனும் மடக்கியின் அடுத்த மறு செய்கைக்கு வேகமாக முன்னேறலாம்.

3.for..of எனும் மடக்கிகள்

இந்த மடக்கியானது ஒப்பீட்டளவில் சமீபத்தியது (2015 இல் அறிமுகப் படுத்தப்பட்டது) மேலும் இது forஎனும் மடக்கியின் எளிமைப்படுத்தப்பட்ட பதிப்பாகும்:

```
const list = [‘a’, ‘b’, ‘c’]

for (const value of list) {

console.log(value) // மதிப்பு

}
```

செயலிகள்(Functions)

ஒரு செயலியானது ஜாவாஉரைநிரலின் மிகமுக்கிய பகுதியாகும். இந்நிலையில் செயலி என்றால் என்ன? எனும் கேள்வி நம்மனைவரின் மனதிலும் எழும் நிற்க.

ஒரு செயலி என்பது தன்னிறைவுடனான ஒரு தொகுப்பான குறிமுறை வரிகளாகும் , பின்வருவது ஒரு செயலியின் அறிவிப்பாகும்:

```
function getData() {

//ஏதாவது செய்

}
```

பின்வரும் கட்டளைவரி போன்ற ஒரு செயலியை எந்த நேரத்திலும் அழைப்பதன் மூலம் செயல்படுத்திடலாம்:

```
getData()
```

ஒரு செயலியானது ஒன்று அல்லது அதற்கு மேற்பட்ட மதிப்புருக்களைக் கொண்டிருக்கலாம்எடுத்துக்காட்டாக:

```
function getData() {
```

```
//ஏதாவது செய்

}

function getData(color) {

//ஏதாவது செய்

}

function getData(color, age) {

//ஏதாவது செய்

}
```

செயலிக்குள் ஒரு மதிப்புருவை கடந்து செல்லவைத்திடும்போது, அந்த செயலியானது கடந்து செல்லும் அளவுருக்களைப் பயன்படுத்தி கொள்கின்றது. எடுத்துக்காட்டாக

```
function getData(color, age) {

//ஏதாவது செய்

}

getData(‘green’, 24)

getData(‘black’)
```

இதில் இரண்டாவது அழைப்பில் black எனும் சரங்களின் அளவுருக்களை color எனும் மதிப்புருவாக

கடந்துசெல்லுமாறுசெய்யப்பட்டுள்ளது, ஆனால் ageஎனும் சரங்களில் அவ்வாறு செய்யப்படவில்லை . இந்த வழக்கில், செயலியின் உள்ளேயே age எனும் மதிப்புரு வரையறுக்கப்பட வில்லை(undefined). இந்த நிபந்தனையைப் பயன்படுத்தி ஒரு மதிப்பானது வரையறுக்கப் படவில்லையாவென சரிபார்க்கலாம்எடுத்துக்காட்டாக :

```
function getData(color, age) {
//ஏதாவது செய்
if (typeof age !== ‘undefined’) {
//…
}
}
```

இதில் typeof என்பது ஓருறுப்பு இயக்கியாகும் இது ஒரு மாறியின் வகையை சரிபார்ப்பதற்காக அனுமதிக்கின்றது இந்த வழிமுறையிலும் நாம் சரிபார்த்திடலாம் எடுத்துக்காட்டாக.

```
function getData(color, age) {
//ஏதாவது செய்
if (age) {
//…
```

}

}

இதில் வயது null என்றும், 0 என்றும் அல்லது வெற்றுசரமாகவும் இருந்தபோதிலும் நிபந்தனையும் உண்மையாக இருக்கின்ற அளவுருக்கள் அனுப்பப்படாவிட்டால், இயல்புநிலை மதிப்புகளை வைத்திருக்கலாம்எடுத்துக்காட்டாக :

```
function getData(color = ‘black’, age = 62) {
//ஏதாவது செய்
}
```

எண்கள், சரங்கள், பூலியன்கள், கோவைகள், பொருள்கள் , செயலிகள் ஆகியவை அனைத்தும் சேர்த்த எந்த மதிப்பையும் அளவுருவாக அனுப்பலாம். ஒரு செயலியானது திருப்பிடுகின்ற மதிப்பு உள்ளது. மதிப்புடன் return எனும் ஒரு திறவுச்சொல்லைச் சேர்க்கும் வரை, இயல்புநிலையாக ஒரு செயலியானது வரையறுக்கப்படாமல் திரும்பும் எடுத்துக்காட்டாக :

```
function getData() {
// ஏதாவது செய்
return ‘hi!’
}
```

ஒரு செயலியை செயல்படுத்தும்போது அது திருப்பும் மதிப்பை ஒரு மாறிக்கு ஒதுக்கீடுசெய்திடலாம் எடுத்துக்காட்டாக.

```
function getData() {
// ஏதாவது செய்
return 'hi!'
}
```

நாம் ஒரு செயலியைஅழைத்திடும்போது இவ்வாறு திருப்பிடும் மதிப்பினை ஒரு மாறிக்கு ஒதுக்கீடு செய்திடலாம் எடுத்துக்காட்டாக.

```
function getData() {
// ஏதாவது செய்
return 'hi!'
}
let result = getData()
result
```

இப்போது result என்பது hi! என்பதுடன் உடன் ஒரு சரத்தை மதிப்பாக வைத்திருக்கிறது! இந்த வழியில் ஒரு மதிப்பை மட்டுமே திரும்பப் பெற முடியும்.பல மதிப்புகளை திருப்ப ஒரு பொருளை அல்லது ஒரு கோவையை பின்வருவதை போன்று திரும்பப் பெறலாம்

```
function getData() {
return [‘Kuppan’, 62]
}
let [name, age] = getData()
```

பிற செயலிகளுக்குள் செயலிகளை வரையறுக்கலாம்எடுத்துக்காட்டாக :

```
const getData = () => {
const dosomething = () => {}
dosomething()
return ‘test’
}
```

உள்ளமைக்கப்பட்ட செயலியை வெளிப்புறஉள்ளடக்கிய செயலியிலிருந்து அழைக்க முடியாது. ஒரு செயலியிலிருந்து வேறொரு செயலியையும் திரும்பப் பெறலாம்.

அம்புக்குறி செயலிகள்(Arrow Functions)

அம்புக்குறி செயலிகள் என்பதுஜாவாஉரைநிரலின் சமீபத்திய அறிமுகமாகும். “வழக்கமான” செயலிகளுக்குப் பதிலாக இவை பெரும்பாலும் பயன்படுத்தப்படுகின்றன.முந்தைய பகுதியில் கூறியவாறான செயலிகள் பயன்படுத்தி கொள்ளப்படுகின்ற எல்லா இடங்களிலும் இந்த அம்புக்குறி செயலிகளையும் பயன்படுத்தி கொள்ளலாம்.கண்ணால் காணும் காட்சியாக, இவ்விரண்டும் குறுகிய இலக்கணத்தின் மூலம் செயலிகளை எழுத அனுமதிக்கின்றன:

```
function getData() {
//…
}
to
() => {
//…
}
```

ஆனால்.. இங்கே பெயர்மட்டும் இல்லை என்பதைக் கவனித்திடுக. ஒரு அம்புக்குறி செயலி அநாமதேயமானதாகும். நாம் அதனை ஒரு மாறிக்கு ஒதுக்கீடு செய்திடலாம். பின்வருவது போன்ற ஒரு மாறிக்கு வழக்கமான செயலியை ஒதுக்கீடுசெய்திடலாம்:

```
let getData = function getData() {
//…
}
```

அவ்வாறு செய்யும்போது, செயலியிலிருந்து பெயரை அகற்றலாம் எடுத்துக்காட்டாக :

```
let getData = function() {
//…
}
```

மேலும் மாறியின் பெயரைப் பயன்படுத்தி செயலியை அழைத்திடலாம் எடுத்துக்காட்டாக:

```
let getData = function() {
//…
}
getData()
```

வழக்கமான செயலிகளை பயன்படுத்திசெயற்படுத்தியதை போன்று அம்புக்குறி செயலிகளிலும் அதே செயல்களை செய்திடலாம் எடுத்துக்காட்டாக:

```
let getData = () => {
//…
}
getData()
```

செயலியின் உட்பகுதியில் ஒரே ஒரு கட்டளை இருந்தால், அடைப்புக் குறிகளைத் தவிர்த்துவிட்டு அனைத்தையும் ஒரே வரியில் எழுதலாம்:

```
const getData = () => console.log('hi!')
```

பொதுவாக அளவுருக்களானவை அடைப்புக்குறிக்குள் கடத்தப் படுகின்றன

```
const getData = (param1, param2) =>
console.log(param1, param2)
```

நம்மிடம் ஒன்று (அல்லது ஒரே ஒரு) அளவுரு இருந்தால், அடைப்புக் குறிகளை முழுவதுமாக தவிர்க்கலாம்:

```
const getData = param => console.log(param)
```

அம்புக்குறி செயலிகள் மறைமுகமாக திரும்பப் பெற நம்மை அனுமதிக்கின்றன: மதிப்புகள் returnஎனும் திறவுச்சொல்லைப்

பயன்படுத்தாமலேயே வழங்கப்படுகின்றன. செயலியின் அமைப்பில் நேரடி இணையத்தின் கூற்று ஒன்று இருக்கும்போது இது நன்றாக செயல்படுகின்றது

```
const getData = () => 'test'

getData() // இது 'test'எனும் முடிவை தருகின்றது
```

வழக்கமான செயலிகளைப் போன்றே, இயல்புநிலை அளவுருக்களைக் கொண்டிருக்கலாம்: அளவுருக்கள் கடத்தப்படாவிட்டால், இயல்புநிலை மதிப்புகளை வைத்திருக்கலாம்:

```
const getData = (color = 'black',

age = 2) => {

//ஏதாவது செய்

}
```

மேலும் நாம் ஒரு மதிப்பை மட்டுமே திரும்பப் பெற முடியும். அம்புக்குறி செயலிகளில் மற்ற அம்புக்குறிசெயலிகள் அல்லது வழக்கமான செயலிகள்போன்று இருக்கலாம்.அவை வழக்கமான செயலிகளுடன் மிகவும் ஒத்தவைகளாக இருக்கும்போது, அவை ஏன் அறிமுகப்படுத்தப்பட்டன? என்ற ஒருகேள்வி இந்நிலையில் நம்மனதில் எழும் நிற்க ,

அவை பொருள் சார்ந்து வழிமுறைகளாகப் பயன்படுத்தப்படும்போது மட்டும். வழக்கமான செயலிகளுடன் பெரியஅளவில் வேறுபடுகின்றன என்பதேயாகும்.இதுவிரைவில் கவனிக்கவேண்டிய ஒரு செய்தியாகும்

19.1 பொருட்கள்(Objects)

ஒரு சரம், ஒரு எண், ஒரு பூலியன், ஒரு சின்னம், பூஜ்யம் அல்லது வரையறுக்கப்படாதது ஆகிய பழமையான வகை இல்லாத எந்தவொரு மதிப்பும் ஒரு பொருளாகும். ஒரு பொருளை எப்படி வரையறுப்பது என்பதற்கான வரையறை பின்வருமாறு:

```
const car = {

}
```

இது ஒரு பொருளிற்கான இலக்கணமாகும், இது ஜாவாஉரைநிரலில் உள்ள நல்ல செய்திகளில் ஒன்றாகும்.new Objectஎனும் தொடரியலையும் பயன்படுத்தலாம்:

```
const car = new Object()
```

இந்நிலையில் Object.create() என்பதைப் பயன்படுத்துவது மற்றொரு தொடரியலாகும்:

```
const car = Object.create()
```

பெரிய எழுத்துடன் செயலிக்கு முன் newஎனும் திறவுச்சொல்லைப் பயன்படுத்தி ஒரு பொருளை துவக்கநிலை யாக்கலாம்(initialize). இந்த செயலியானது அந்த பொருளுக்கு ஒரு கட்டமைப்பாளராக செயல் படுகிறது. அங்கு, பொருளின் துவக்க நிலையைஅமைக்க,

அளவுருக்களாக நாம் பெறும் தருக்கங்களை துவக்கநிலை யாக்கலாம்(initialize).

```
function Car(brand, model) {
this.brand = brand
this.model = model
}
```

ஒரு new எனும் பொருளைப் பயன்படுத்தி துவக்கநிலையாக்கலாம்

```
const myCar = new Car('Ford', 'Fiesta')
myCar.brand //' Fordஎனும் மாதிரிவகையானதாகும்'
myCar.model //'Fiesta'எனும் மாதிரி வகையானதாகும்
```

பொருட்கள் எப்போதும் மேற்குறிப்பு மூலம் கடத்தப்படுகின்றன.ஒரு மாறிக்கு மற்றொன்றின் அதே மதிப்பை ஒதுக்கினால், அது ஒரு எண் அல்லது சரம் போன்ற பழமையான வகையாக இருந்தால், அவை மதிப்பின் மூலம் அனுப்பப்படுகின்றன:பின்வரும் எடுத்துகாட்டினை கருத்தில்கொள்க:

```
let age = 61
let myAge = age
myAge = 62
```

```
age // இங்கு வயது 61ஆகும்

const car = {

color: ‘blue’

}

const anotherCar = car

anotherCar.color = ‘yellow’

car.color //‘மஞ்சள் வண்ணமாகும் yellow’
```

கோவைகள் அல்லது செயலிகள் கூட, hoods என்பதன் கீழ், பொருட்களாகும், எனவே அவை எவ்வாறு செயல்படுகின்றன என்பதைப் புரிந்துகொள்வது மிகவும் முக்கியமாகும்.

19.2. பொருளின் பண்பியல்புகள்Object properties

பொருட்களுக்கு பண்பியலபுகள் உள்ளன, அவை மதிப்புடன்தொடர்புடைய அடையாளத்தால் உருவாக்கப்படுகின்றன.ஒரு பண்பியல்பின் மதிப்பு எந்த வகையிலும் இருக்கலாம், அதாவது அது ஒரு கோவையாக இருக்கலாம், ஒரு செயலியாக இருக்கலாம்.மேலும் அது ஒரு பொருளாக கூட இருக்கலாம், ஏனெனில் பொருட்கள் மற்ற பொருட்களில் உள்ளமைக்கப் பட்டு இருக்க முடியும். முந்தைய பகுதியில் நாம் பார்த்துவந்த பொருளிற்கான மதிப்புரு இலக்கணம் இதுவேயாகும் எடுத்துகாட்டாக:

```
const car = {
}
```

இதில்colorஎன்பதன் பண்பியல்புகளை பின்வருமாறு வரையறுக்கலாம்:

```
const car = {
color: ‘blue’
}
```

இங்கே நம்மிடம் ஒரு car எனும் ஒருபொருள் உள்ளதுஎன்க, அதன் வண்ண மதிப்பு நீல நிறத்தில் உள்ளது. அடையாளங்கள் எந்த சரமாகவும் இருக்கலாம், ஆனால் சிறப்பு எழுத்துகள் குறித்து எச்சரிக்கையாக இருந்திடுக: பண்பியல்பின் பெயரில் ஒரு மாறிப் பெயராக செல்லுபடியாகாத எழுத்தைச் சேர்க்க விரும்பினால், அதைச் சுற்றி மேற்கோள்களைப் பயன்படுத்த வேண்டியிருக்கும்எடுத்துகாட்டாக :

```
const car = {
color: 'blue',
'the color': 'blue'
}
```

செல்லுபடியாகாத மாறியின் பெயர் எழுத்துக்களில் இடைவெளிகள், இடைஅடிக்கோடுகள், பிற சிறப்பு எழுத்துக்கள் ஆகியவைகளும்அடங்கும்

நாம் காண்பது போன்று, நம்மிடம் பல பண்பியல்புகள் இருக்கும்போது, ஒவ்வொரு பண்பியல்பினையும் கால்புள்ளியால் பிரிக்கலாம். இரு வெவ்வேறு இலக்கணங்களைப் பயன்படுத்தி ஒரு பண்பியல்பின் மதிப்பை நாம் மீட்டெடுக்கலாம். முதலாவது புள்ளி குறிவரி பின்வருமாறு

car.color //‘நீலவண்ணம்’

இரண்டாவது (செல்லுபடியாகாத பெயர்களைக் கொண்ட பண்பியல்பு களுக்கு நாம் பயன்படுத்தக்கூடிய ஒரேயொரு), பின்வருமாறு பகர அடைப்புக் குறிகளைப் பயன்படுத்துவதாகும்:

car[‘the color’] // நீல வண்ணம்

இல்லாத பண்பியல்பினைஅனுகினால், undefinedZ எனும் மதிப்பைப் பெறுவோம் எடுத்துகாட்டாக:

car.brand //வரையறுக்கப்படாதது

மேலே கூறியது போன்று, பொருட்கள் உள்ளமைந்த பொருட்களின் பண்பியல்புகளாகக் கொண்டிருக்கலாம் எடுத்துகாட்டாக:

```
const car = {
brand: {
name: ‘Ford’
},
color: ‘blue’
}
```

இந்த எடுத்துக்காட்டில், வணிகச்சின்னத்தின் பெயரைப் பயன்படுத்தி அணுகலாம் எடுத்துகாட்டாக

car.brand.name

or

car[‘brand’][‘name’]

பொருளை வரையறுக்கும்போது அதனுடைய பண்பியல்பின் மதிப்பை அமைக்கலாம். ஆனால் எப்போது வேண்டுமானாலும் அதை பின்னர் புதுப்பிக்கலாம் எடுத்துகாட்டாக:

const car = {

color: ‘blue’

}

car.color = ‘yellow’

car[‘color’] = ‘red’

மேலும் ஒரு பொருளுக்கு புதிய பண்பியல்புகளையும் சேர்க்கலாம் எடுத்துகாட்டாக.

car.model = ‘Fiesta’

car.model //’Fiesta’எனும் வணிகச்சின்னத்துடன்

பொருளை பின்வருமாறு கொடுக்கலாம்

const car = {

color: ‘blue’,

brand: ‘Ford’

}

இதைப் பயன்படுத்தி இந்தப் பொருளிலிருந்து ஒரு பண்பியல்பினை பின்வருமாறு நீக்கம் செய்திடலாம்

delete car.brand

19.3.பொருளின் வழிமுறைகள்(Object methods)

முந்தைய பகுதிகளில் செயலிகளைப் பற்றி பார்த்துவந்தோம். செயலிகளுக்கு ஒரு செயலியின் பண்பியல்பினைக்கு ஒதுக்கலாம், இந்த செயலில், அவை வழிமுறைகள் என்று அழைக்கப்படுகின்றன. பின்வரும் எடுத்துக்காட்டில், தொடக்கப் பண்பியல்புக்கு ஒரு செயலி ஒதுக்கப்பட்டுள்ளது, மேலும் இறுதியில் அடைப்புக்குறிகளுடன், பண்பியல்புகளுக்குப் பயன்படுத்திய திறுத்தற்குறி . எனும் புள்ளியின் இலக்கணத்தின் மூலம் அதை செயல்படுத்தலாம்:

```
const car = {
brand: ‘Ford’,
model: ‘Fiesta’,
start: function() {
console.log(‘Started’)
}
}
car.start()
```

ஒரு function() {} எனும் தொடரியலை பயன்படுத்தி வரையறுக்கப்பட்ட ஒரு வழிமுறையின் உள்ளே இதை

குறிப்பிடுவதன் மூலம் பொருள் நிகழ்வை அணுகலாம். பின்வரும் எடுத்துக்காட்டில்,this.brand ,this.modelஎன்றவாறு இவைகளைப் பயன்படுத்தி brand model ஆகியவற்றின் பண்பியல்புகளின் மதிப்புகளை அனுகலாம் எடுத்துகாட்டாக

```
:const car = {
brand: ‘Ford’,
model: ‘Fiesta’,
start: function() {
console.log(`Started
${this.brand} ${this.model}`)
}
}
car.start()
```

வழக்கமான செயலிகளுக்கும் அம்புக்குறி செயலிகளுக்கும் இடையிலான இந்த வேறுபாட்டைக் குறிப்பிடுவது மிகமுக்கியமாகும்: இந்நிலையில் அம்புக்குறி செயலியைப் பயன்படுத்தினால், இதை அனுக முடியாது:

```
const car = {
brand: ‘Ford’,
model: ‘Fiesta’,
```

```
start: () => {
console.log(`Started
${this.brand} ${this.model}`) // இது செயல்படாது
}
}
car.start()
```

அம்புக்குறி செயலிகள் பொருளுடன் பிணைக்கப்படவில்லை என்பதே இதற்குக் காரணமாகும்.வழக்கமான செயலிகள் பெரும்பாலும் பொருள் வழிமுறைகளாகப் பயன்படுத்தப்படுவதற்கான காரணம் இதுதான். வழிமுறைகள் வழக்கமான செயலிகள் போன்ற அளவுருக்களை ஏற்கலாம் எடுத்துகாட்டாக:

```
const car = {
brand: ‘Ford’,
model: ‘Fiesta’,
goTo: function(destination) {
console.log(Going to ${destination})
}
}
car.goTo(‘Rome’)
```

இனங்கள்(Classes)

ஜாவாஉரைநிரலின் மிகவும் சுவாரஸ்யமான பகுதிகளில் ஒன்றான பொருட்களைப் பற்றி இதுவரை பார்த்துவந்தோம்.இந்த பகுதியில் இனங்களை அறிமுகப்படுத்திடும்நிலைக்குசிறிது முன்னேறிச் செல்வோம்.

இனங்கள் என்றால் என்ன?

அவை பல பொருட்களுக்கான பொதுவான வடிவத்தை வரையறுப் பதற்கான ஒரு வழியாகும் பின்வருமாறு எடுத்துகாட்டிலுள்ளதைபோன்று personஎனும் ஒரு பொருளை எடுத்துக் கொள்க:

```
const person = {
name: ‘Kuppan’
}
```

Person என்ற பெயரில் ஒரு இனத்தினை உருவாக்கலாம் (P என்பது பெரிய எழுத்தாக இருப்பதை கவனத்தில்கொள்க, இனங்களைப் பயன் படுத்தும் போது ஒரு மரபு

பின்பற்றுவதையும்கவனத்தில்கொள்க), அதற்கு ஒரு name எனும் பண்பியல்பு உள்ளது எடுத்துகாட்டாக:

```
class Person {

name

}
```

இப்போது இந்த இனத்திலிருந்து, **Kuppan** என்பது போன்ற ஒரு பொருளைத் துவக்குவதாக கொள்க எடுத்துகாட்டாக:

```
const Kuppan = new Person()
```

இதில்Kuppan என்பது ஒரு Person எனும் இனத்தின் நிகழ்வு என்று அழைக்கப் படுகிறது. பின்னர் இதற்கு name எனும் பண்பியில்பின் மதிப்பை அமைக்கலாம் எடுத்துகாட்டாக.

```
Kuppan.name = ’ Kuppan
```

மேலும் அதை பயன்படுத்தி பின்வருமாறு அனுக முடியும்

```
Kuppan.name
```

பொருளின்பண்பியல்புகளுக்கு செய்வது போன்று இனங்களின் name,எனும் வழி முறைகள் போன்ற பண்பியல்புகளை வைத்திருக்க முடியும்.பொதுவாக வழிமுறைகள் இவ்வாறான வழியிலேயே வரையறுக்கப் படுகின்றன எடுத்துகாட்டாக :

```
class Person {
```

```
hello() {
return ‘Hello, I am Kuppan’
}
}
```

இனத்தின் ஒரு நிகழ்வில் வழிமுறைகளைஅழைக்க முடியும் எடுத்துகாட்டாக :

```
class Person {
hello() {
return ‘Hello, I am Kuppan’
}
}
const flavio = new Person()
Kuppan.hello()
```

இதில் constructor() எனப்படும் ஒரு சிறப்பு வழிமுறையும் உள்ளது, அதை நாம் ஒரு புதிய பொருளின் நிகழ்வை உருவாக்கும் போது இனத்தின் பண்பியல்புகளை துவக்கநிலையாக்குவதற்காக(initialize) பயன்படுத்தலாம். இது பின்வருமாறு செயல்படுத்தபடுகிறது

```
class Person {
constructor(name) {
this.name = name
```

```
}
hello() {
return ‘Hello, I am’ + this.name + ‘.’
}
}
```

பொருள் நிகழ்வை அனுக thisஎன்பதை எவ்வாறு பயன்படுத்தப்படுகிறது என்பதைக் கவனித்துகொள்க.இப்போது இனத்தில் இருந்து ஒரு புதிய பொருளை உடனுக்குடன், ஒரு சரத்தை கடந்து, hello என்று அழைக்கும் போது, தனிப்பயனாக்கப்பட்ட செய்தியைப் பெறுலாம்எடுத்துகாட்டாக :

```
const Kuppan = new Person(‘Kuppan’)
Kuppan.hello() //‘ஐயா, நான் Kuppan.எனும் பெயர்கொண்டவனாகும்’
```

பொருளானது துவக்கநிலையாக்கிடும்போது(initialize) constructor எனும் வழிமுறையானது கடந்து செல்கின்ற எந்தவொரு அளவுருக்களுடன் அழைக்கப்பெறுகின்றன.பொதுவாக வழிமுறைகள் பொருள் நிகழ்வில் வரையறுக்கப்படுகின்றன, இனத்தில் அல்ல.அதற்கு பதிலாக இனத்தில் செயல்படுத்த அனுமதிக்கின்ற ஒரு முறையை static என வரையறுக்கலாம் எடுத்துக்காட்டாக:

```
class Person {
```

```
static genericHello() {

return ‘Hello’

}

}

Person.genericHello() //Hello
```

இது சில நேரங்களில் மிகவும் பயனுள்ளதாக இருக்கும்

மரபுரிமை (Inheritance)

ஒரு இனமானது மற்றொரு இனத்துடன் நீட்டிக்க முடியும், மேலும் அந்த இனத்தினைப் பயன்படுத்தி துவக்கநிலையாக்கப்படும்(initialize) பொருள்கள் இரு இனங்களின் அனைத்து வழிமுறைகளையும் பெறுகின்றன.நம்மிடம் ஒருஇன Person ஆனது பின்வருவதை போன்று இருப்பதாகக்கொள்க:

```
class Person {
hello() {
return ‘Hello, I am a Person’
}
}
```

இதில்Personஐ நீட்டிக்கும் புதிய இனத்திற்கான நிரலாளரை வரையறுக்கலாம் எடுத்துக்காட்டாக.

```
class Programmer extends Person {
}
```

இப்போது நிரலாளர் எனும் இனத்தின் மூலம் ஒரு புதிய பொருளை உடனடியாக உருவாக்கினால், அது hello() எனும் வழிமுறையை அணுகிடுகின்றது எடுத்துக்காட்டாக

```
const.Kuppan = new Programmer()
Kuppan.hello() //'ஐயா நான் ஒரு மனிதன்'
```

குழந்தை இனத்திற்குள், super() என அழைக்கும் பெற்றோர் இனத்தினைள் குறிப்பிடலாம்எடுத்துக்காட்டாக :

```
class Programmer extends Person {
hello() {
return super.hello() +
'. I am also a programmer.'
}
}
const.Kuppan = new Programmer()
Kuppano.hello()
```

மேலே உள்ள நிரலை செயல்படுத்தினால் Hello, I am a Person .I am also a programmerஎன அச்சிடுகிறது.

ஒத்திசைவற்ற(Asynchonous)நிரலாக்கமும்மீளழைப்புகளும்(callbacks)

பெரும்பாலான நேரங்களில், ஜாவாஉரைநிரலின் குறிமுறைவரிகள் ஒத்திசைவாக இயங்குகின்றன. இதன் பொருள் குறிமுறைவரிகளில் முதலில் ஒரு குறிமுறைவரி செயல்படுத்தப்படுகிறது,பின்னர்அடுத்தடுத்த குறிமுறைவரிகள் தொடர்ந்து செயல்படுத்தப்படுகின்றன,என்பதாகும் நாம் எதிர்பார்ப்பது போல் எல்லாம் உள்ளது பெரும்பாலான நிரலாக்கங்களில் இவ்வாறுதான் செயல்படுத்துபடுகின்றன.

இருப்பினும், குறிமுறைவரிகளின் குறிப்பிட்டதொரு குறிமுறைவரியை இயக்குவதற்காக காத்திருக்க முடியாத நேரங்களும் உள்ளன. ஒரு பெரிய கோப்பானது நினைவகத்தில் ஏற்றப்பட்டு நிரலை முழுவதுமாக செயலில் நிலைநிறுத்தம் செய்வதற்கு ஓரிரு வினாடிகூட காத்திருக்க முடியாமல் நம்மில் பெரும்பாலானோர் தவித்திடுவோம். வேறு ஏதாவது செய்வதற்கு

முன்இணையத்திலிருந்து அதற்கான ஆதாரங்கள் பதிவிறக்கப்படும் வரை காத்திருக்க முடியாமல் தவித்திடுவோம்.இவ்வாறான சிக்கல்களை ஜாவாஉரைநிரல் ஆனதுமீளழைப்புகள்**(callbacks)**எனும் வசதியைப் பயன்படுத்தி தீர்வுசெய்கிறது.

இந்த மீளழைப்புகளைஎவ்வாறு பயன்படுத்துவது என்பதற்கான எளிய எடுத்துக்காட்டுகளில் ஒன்று காலங்காட்டிகள்**(**timers**)**ஆகும். காலங் காட்டிகள் ஜாவாஉரைநிரலின் பகுதியாக இல்லை, ஆனால் அவை இணைய உலாவி , Node.js ஆகியவற்றின் மூலம் வழங்கப்படுகின்றன. இந்தகாலங்காட்டிகளில் setTimeout()என்பதைப் பற்றி இப்போது காண்போம்:

.setTimeout() எனும் செயலியானது ஒரு செயலி, ஒரு எண் ஆகிய இரு தருக்கங்களை ஏற்றுக் கொள்கிறது: இதில்.எண் என்பது ஒரு செயலி யானது இயங்குவதற்கு முன் கடக்க வேண்டிய மில்லி விநாடிகள் ஆகும்.

எடுத்துகாட்டாக:

```
setTimeout(() => {

// இருவினாடிகளுக்கு பிறகு செயல்படுக

console.log('inside the function')

}, 2000
```

மேலும்console.log(‘inside the function’) எனும் குறிமுறைவரியைக் கொண்ட செயலியானது இரண்டு வினாடிகளுக்குப் பிறகு செயல்படுத்தப்படுகின்றது. இந்த செயலிக்கு முன்பகுதியிலும் console.log(‘before’) என்பதையும்,பின்னர் இதேசெயலிக்கு பின்புறம் console.log(‘after’) என்பதையும் சேர்த்தால் இந்த செயலியானது பின்வருமாறுஅமைந்திருக்கின்றது:

```
setTimeout(() => {
// இருவினாடிகளுக்கு பிறகு செயல்படுக
console.log(‘inside the function’)
}, 2000)
console.log(‘after’)
```

இதனை செயல்படுத்தியபின்னர் முகப்பத்திரையில்பின்வருமாறான விளைவைக் காணலாம்:

```
before
after
inside the function
```

மீளஅழைக்கும் செயலியானது ஒத்திசைவற்ற முறையில் செயல்படுத்தப் படுகிறது. கோப்பு முறைமை, வலைபின்னல்கள், நிகழ்வுகள் அல்லது இணைய உலாவியில் DOM உடன் பணிபுரியும்

போது இது மிகவும் பொதுவான வடிவமாகும்.இங்கு குறிப்பிட்டுள்ள எல்லா செய்திகளும் “core” ஜாவாஉரைநிரல் அல்ல என்ற செய்தியை மனதில் கொள்க, எனவே அவை இந்த கையேட்டில் விளக்கப்படவில்லை,

ஒரு callback அளவுருவை ஏற்கும் செயலியை வரையறுத்திடுலாம், இது ஒரு செயலியாகும். இது குறிமுறைவரிகளை திரும்ப அழைப்பதற்குத் தயாராக இருக்கும் போது, முடிவைக் கடப்பதற்குமுன் அதை அழைக்கின்றது எடுத்துக்காட்டாக

```
const doSomething = callback => {

//செயல்களைசெய்

//செயல்களைசெய்

const result = /* .. */

callback(result)

}
```

இந்த செயலியைப் பயன்படுத்துகின்ற குறிமுறைவரிகளை பின்வருமாறும் பயன்படுத்திடலாம்:

```
doSomething(result => {

console.log(result)

})
```

ஒழுங்கினங்கள்(Promises)

ஒத்திசைவற்ற குறிமுறைவரிகளைக் கையாளுவதற்கான மாற்று வழி ஒழுங்கினங்களாகும். முந்தைய பகுதியில் கண்டதை போன்று, callbacksஎன்பதன் மூலம், ஒரு செயலியை மற்றொரு செயலியின் அழைப்பிற்கு கடத்தும்போது, அது அந்த செயலியின் செயலாக்கத்தை முடித்ததும் . பின்வருவது போன்று அழைக்கின்றது:

```
doSomething(result => {
console.log(result)
})
```

இதில் doSomething() எனும் செயலியானது குறிமுறைவரிகளின் செயல்பாடு முடிவடையும் போது, பெறப்பட்ட செயலியை ஒரு அளவுருவாக அழைக்கிறது எடுத்துக்காட்டாக:

```
const doSomething = callback => {
//செயல்களைசெய்
//செயல்களைசெய்
const result = /* .. */
```

```
callback(result)
}
```

இந்த அனுகுமுறையின் முக்கிய பிரச்சனை என்னவென்றால், இந்த செயலியின் முடிவை நமது மீதமுள்ள குறிமுறைவரிகளில் பயன்படுத்த வேண்டும் எனெனில், நம்முடைய எல்லா குறிமுறைவரிகளும் callbackஎனும் செயலிக்குள் உள்ளமைக்கப்பட வேண்டும், மேலும் 2-3 மீளஅழைப்புகள் செய்ய வேண்டியிருந்தால், பொதுவாக “callback hell” என வரையறுத்து பல நிலை செயலிகளுடன் பிற செயலிகளில் உள்தள்ளப்பட்டு அதை உள்ளிடவேண்டும்.எடுத்துக்காட்டாக:

```
doSomething(result => {
doSomethingElse(anotherResult => {
doSomethingElseAgain(yetAnotherResult => {
console.log(result)
})
})
})
```

அவ்வாறு செய்வதற்கு பதிலாக இதை சமாளிக்க ஒழுங்கினம்(Promise) எனும் செயலி சிறந்தவொரு வழியாகும் எடுத்துக்காட்டாக .:

```
doSomething(result => {
console.log(result)
})
```

ஒழுங்கினத்தின் அடிப்படையிலான செயலியை பின்வருமாறான வழியில் அழைக்கலாம்:

```
doSomething()
.then(result => {
console.log(result)
})
```

முதலில் செயலியை அழைக்கின்றது, பின்னர் then() எனும் செயலியானது செயல் முடிவடையும் போது அழைக்கப்படுகிறது. உள்தள்ளல் ஒரு பொருட்டல்ல, ஆனால் தெளிவுக்காக இந்த பாணியை அடிக்கடி பயன்படுத்துவது நல்லது.
பொதுவாகcatch()எனும் வழிமுறையைப் பயன்படுத்தி பிழைகளைக் கண்டறிவது நல்லது எடுத்துக்காட்டாக :

```
doSomething()
.then(result => {
console.log(result)
})
.catch(error => {
```

```
console.log(error)
})
```

இப்போது, இந்த இலக்கணத்திினை பயன்படுத்த, doSomething() செயலியை செயல்படுத்துவது கொஞ்சம் சிறப்பு வாய்ந்ததாக இருக்க வேண்டும். அதாவது அதை ஒரு சாதாரண செயலியாக அறிவிப்பதற்கு பதிலாக Promises API என்றவாறு பயன்படுத்துவது நல்லது எடுத்துக்காட்டாக:

```
const doSomething = () => {
}
```

இதை ஒரு ஒழுங்கினப் பொருளாக அறிவிக்கலாம்எடுத்துக்காட்டாக :

```
const doSomething = new Promise()
```

மேலும் ஒழுங்கினத்தின் கட்டமைப்பாளரில் ஒரு செயலியை கடத்தலாம் எடுத்துக்காட்டாக

```
const doSomething = new Promise(() => {
})
```

இதில்முதலாவது ஒழுங்கினத்திினைத் தீர்வுசெய்வதற்காக(resolve)எனும் அழைக்கும் செயலி, இரண்டாவது ஒழுங்கினத்திினை நிராகரிப்ப தற்காக (reject) எனும் செயலி.ஆகிய இருஅளவுருக்களைப் பெறுகிறது எடுத்துக்காட்டாக.

```
const doSomething = new Promise(
(resolve, reject) => {
})
```

ஒரு ஒழுங்கினத்தினைத் தீர்வுசெய்வது என்பது அதை வெற்றிகரமாக முடிப்பதாகும் (அதை யார் பயன்படுத்துகிறார்கள் என்பதில் then() எனும் வழிமுறையை அழைப்பதில் தீர்வு உருவாகின்றது).ஒரு ஒழுங்கி னத்தினை நிராகரிப்பது என்பது பிழையுடன் முடிப்பதாகும் (அதை யார் பயன்படுத்து கிறார்கள் என்பதில் catch() எனும் வழிமுறையை அழைப்பதில் செயல்படுத்தப் படுகின்றது.அவை பின்வருமாறு:

```
const doSomething = new Promise(
(resolve, reject) => {
//சில குறிமுறைவரிகள்
const success = /* … */
if (success) {
resolve(‘ok’)
} else {
reject(‘this error occurred’)
}
}
```

)

நாம் விரும்பும் எந்த வகையிலான செயலிகளையும் தீர்வு செய்திடவும் நிராகரிக்கவும் ஒரு அளவுருவை அனுப்பலாம்.

ஒத்திசைவும்(Async)காத்திருத்தலும்(Await)

Async எனும் செயலி ஒழுங்கினம்(Promise) எனும் செயலியை விட உயர் மட்ட நிலையில் சுருக்கமான செயலாகும். பின்வரும் எடுத்துக்காட்டில் உள்ளதைப் போன்று ஒரு ஒத்திசைவு செயலியானது ஒரு ஒழுங்கினத்தினை வழங்குகிறது:

```
const getData = () => {
return new Promise((resolve, reject) => {
setTimeout(() =>
resolve('some data'), 2000)
})
}
```

இந்தச் செயலியைப் பயன்படுத்த விரும்பும் எந்தக் குறிமுறைவரியும் செயலுக்கு முன்asyncஎனும் திறவு சொல்லைப் பயன்படுத்துகின்றது எடுத்துக்காட்டாக

```
const data = await getData()
```

அவ்வாறு செய்தால், ஒழுங்கினம் எனும் வசதியின் மூலம் வழங்கப்படு கின்ற எந்தத் தரவும் dataஎனும் மாறிக்கு ஒதுக்கப்படுகின்றது. இங்கு, data என்பது “some data” எனும் சரங்களாகும். ஒரு குறிப்பிட்ட எச்சரிக்கையுடன்: நாம் await எனும் திறவுச்சொல்லைப் பயன்படுத்தும் போதெல்லாம்,async என வரையறுக்கப்பட்ட ஒரு செயலிக்குள் அதை பின்வருவதை போன்று செய்ய வேண்டும்.

```
:const doSomething = async () => {

const data = await getData()

console.log(data)

}
```

இதில் Async/waiit ஆகிய இரட்டை செயலிகளானவை,சுத்தமான குறிமுறை வரிகளையும், ஒத்திசைவற்ற குறிமுறைவரிகளுடன் பணிபுரிய மனத்தின் எளிய மாதிரியையும் பெற அனுமதிக்கிறது. மேலே உள்ள எடுத்துக்காட்டில் உள்ள குறிமுறைவரிகள் மிகவும் எளிமையானவை. promises, அல்லதுcallback ஆகிய செயலிகளைப் பயன்படுத்திய குறிமுறைவரிகளுடன் ஒப்பீடு செய்து சரிபார்த்திடுக. இது மிகவும் எளிமையான உதாரணமாகும், குறிமுறைவரிகள் மிகவும் சிக்கலானதாக இருக்கும்போது பல்வேறு முக்கிய நன்மைகள் கிடைக்கின்றன.

எடுத்துகாட்டக, Fetch API ஐப் பயன்படுத்தி JSON ஆதாரத்தைப்

பெறுவது promisesஐப் பயன்படுத்தி அதை பாகுபடுத்திடுவது எவ்வாறு என பின்வரும் எடுத்துகாட்டின் வாயிலாக அறிந்துகொள்க.

```
const getFirstUserData = () => {

// பயனாளர் பட்டியலைப் பெறுக

return fetch('/users.json')

// JSON ஐ பாகுபடுத்திடுக

.then(response => response.json())

// முதல் பயனாளரைத் தேர்ந்தெடுத்திடுக

.then(users => users[0])

// பயனாளரின் தரவை பெறுக

.then(user =>

fetch(/users/${user.name}))

// JSON ஐ பாகுபடுத்திடுக

.then(userResponse => response.json())

}

getFirstUserData()
```

await/asyncஐப் பயன்படுத்தி வழங்கப்படும் அதே செயலிபின்வருமாறு:

```
const getFirstUserData = async () => {

// பயனாளரின் பட்டியலைபெறுக

const response = await fetch('/users.json')

// JSON ஐ பாகுபடுத்திடுக

const users = await response.json()

// முதல் பயனாளரை தெரிவுசெய்திடுக

const user = users[0]

// பயனாளரின் தரவை பெறுக

const userResponse =

await fetch(/users/${user.name})

// JSON ஐ பாகுபடுத்திடுக

const userData = await user.json()

return userData

}

getFirstUserData()
```

மாறிகளின் செயற்பரப்பு (Variables scope)

முந்தைய பகுதிகளில் மாறிகளை அறிமுகப்படுத்தியபோதுconst , let , var ஆகியவற்றை எவ்வாறு பயன்படுத்துவது என்பதைப் பற்றி கண்டு வந்தோம். இப்போது Scope என்பது குறித்து காண்போம் Scope என்பது நிரலின் ஒரு பகுதிக்குத் தெரிகின்ற மாறிகளின் தொகுப்பாகும். ஜாவாஉரைநிரலில் உலகளாவிய செயற்பரப்பு(global scope),தொகுப்பிற்குள் மட்டுமான செயற்பரப்பு(block scope),செயலிக்குள் மட்டுமான செயற்பரப்பு (function scope) ஆகியவை உள்ளன.

பொதுவாக ஒரு செயலிக்கு அல்லது தொகுப்புக்கு வெளியே ஒரு மாறி வரையறுக்கப்பட்டால், அது உலகளாவிய பொருளுடன் இணைக்கப் பட்டுள்ளதுஎன்றும் global scope ஐ கொண்டுள்ளது என்றும் பொருளாகும், அதாவது குறிப்பிட்ட அந்த செயலியானது ஒரு நிரலின் அனைத்து பகுதியிலும் கிடைக்கிறது என பொருள்கொள்ளலாம்.

var, let, const ஆகியவற்றின் அறிவிப்புகளுக்கு இடையே மிக

முக்கியமான வேறுபாடு உள்ளது.ஒரு செயலியின் உள்ளே var என வரையறுக்கப்பட்ட ஒரு மாறி அந்த செயலிக்குள் மட்டுமே தெரியும்.மேலும் செயலியின் மதிப்புருக்களும் அதே போன்றதே யாகும்:

const அல்லது let என வரையறுக்கப்பட்ட ஒரு மாறி, அது வரையறுக்கப் பட்ட தொகுப்பிற்குள் மட்டுமே தெரியும்.

தொகுப்பு(block) என்பது { } என்றவாறான ஒரு ஜோடியான இருதலை அடைப்பு குறிகளுக்குள் தொகுக்கப்பட்ட வழிமுறைகளாகும்

if எனும் கூற்று அல்லது forஎனும் மடக்கியின் ஒரு செயலிகூட இதேபோன்ற தொகுப்பின் உள்பகுதிக்குள் மட்டும் இருப்பதை காணலாம்.

இந்நிலையில்ஒரு தொகுப்பு ஆனது var க்கு ஒரு புதிய செயற்பரப்பினை வரையறுக்கவில்லை, ஆனால் அது let, const ஆகிய இரண்டையும் வரையறுக்கிறது என்பதைப் புரிந்துகொள்வது மிகமுக்கியமாகும்.

இது மிகவும் நடைமுறை தாக்கங்களைக் கொண்டுள்ளது.ஒரு செயலியில் நிபந்தனைக்குட்பட்ட ஒரு var எனும் மாறியை வரையறுப்பதாக கொள்க எடுத்துக்காட்டாக.

```
function getData() {
if (true) {
var data = ‘some data’
```

```
console.log(data)
}
}
```

இந்த செயலியை அழைத்தால், முகப்புத்திரையில் some data என அச்சிடுவதை காணலாம்

இதில்if க்குப் பிறகு console.log(data) ஐ நகர்த்த முயற்சித்தால், அது இன்னும் செயல்படுகிறது எடுத்துகாட்டாக:

```
function getData() {
if (true) {
var data = 'some data'
}
console.log(data)
}
```

ஆனால் var dataஎன்பதைlet data என மாற்றினால் எடுத்துகாட்டாக:

```
function getData() {
if (true) {
let data = 'some data'
}
console.log(data)
```

}

உடன் திரையில் ReferenceError: data is not defined எனும் ஒரு பிழைச்செய்தி தோன்றிடுவதை காணலாம் ஏனெனில் var எனும் செயலியானது பரந்த செயற்பரப்பினை கொண்டது, மேலும் இது ஒரு சிறப்பானதென உயர்த்திடுகின்றது . சுருக்கமாக, குறிமுறைவரிகளை இயக்கும் முன், ஜாவாஉரைநிரல் மூலம் var அறிவிப்பு மிக நெருக்கமான செயலியின் மேல் நகர்த்தப்படுகிறது. அதிகமாகவோ அல்லது குறைவாகவோ இந்த செயலியானது JS க்கு உள்ளக்கத்திற்குள் தெரிகிறதுஎடுத்துகாட்டாக :

```
function getData() {
var data
if (true) {
data = ‘some data’
}
console.log(data)
}
```

அதனால்தான், ஒரு செயலியின் மேல்பகுதியில் console.log(data) என்பது அறிவிக்கப்படுவதற்கு முன்பே, அந்த மாறிக்கான மதிப்பாக வரையறுக்கப் படாமல் இருக்கின்றது எடுத்துகாட்டாக:

```
function getData() {
```

```
console.log(data)
if (true) {
var data = ‘some data’
}
}
```

ஆனால் இங்கு let என்பதற்கு மாறினால், உடன் ReferenceError: data is not defined எனும் ஒரு பிழைசெய்தி திரையில் தோன்றுகின்றது ஏனெனில் உயர்த்துகின்ற செயல் letஎன்பதன்அறிவிப்புகளை அனுமதிக்காது. const என்பதற்கும் letஎன்பதனுடைய அதே விதிகளைப் பின்பற்றுகிறது: இதுவே தொகுப்பின் செயற்பரப்பாகும். முதலில் இது தந்திரமானதாக இருக்கலாம், ஆனால் இந்த வித்தியாசத்தை உணர்ந்தவுடன், letஎன்பதுடன் ஒப்பிடும்போது var ஏன் மோசமான நடைமுறையாகக் கருதப்படுகிறது என்பதை காணலாம்: அவை குறைவான நகரும் பகுதிகளைக் கொண்டுள்ளன, மேலும் அவற்றின் நோக்கம் தொகுப்பிற்குள் மட்டுமென மட்டுப்படுத்தப் பட்டுள்ளது. அவை மடக்கி மாறிகளாக இருப்பது மிகவும் நல்லது, ஏனெனில் ஒரு மடக்கியின் செயல் முடிந்ததும் அவை இல்லாமல் போகும் எடுத்துகாட்டாக:

```
function doLoop() {
for (var i = 0; i < 10; i++) {
console.log(i)
```

```
}
console.log(i)
}
doLoop()
```

இதில் மடக்கியில் இருந்து வெளியேறும் போது, i என்பதன் மதிப்பு 10 உடன் செல்லுபடியாகும் மாறியாக இருக்கின்றது. இந்நிலையில் let என்பதற்கு மாறியபின்னர், console.log(i) என்பதை செயல்படுத்த முயற்சித்தால் ReferenceError: i is not definedஎன்ற பிழை செய்தி கிடைக்கின்றது

முடிவுரை

இதுவரை பொறுமையாக இந்த துவக்கநிலையாளர்களுக்கான ஜாவாஉரைநிரல் கையேட்டினை படித்ததற்கு மிக்க நன்றி.

ஜாவாஉரைநிரல் பற்றி அடிப்படையாக அறிய இது ஊக்குவிக்ககூடும் என்று நம்புகிறேன்.

FREETAMILEBOOKS.COM

மின்புத்தகங்களைப் படிக்க உதவும் கருவிகள்:

மின்புத்தகங்களைப் படிப்பதற்கென்றே கையிலேயே வைத்துக் கொள்ளக்கூடிய பல கருவிகள் தற்போது சந்தையில் வந்துவிட்டன. Kindle, Nook, Android Tablets போன்றவை இவற்றில் பெரும்பங்கு வகிக்கின்றன. இத்தகைய கருவிகளின் மதிப்பு தற்போது 4000 முதல் 6000 ரூபாய் வரை குறைந்துள்ளன. எனவே பெரும்பான்மையான மக்கள் தற்போது இதனை வாங்கி வருகின்றனர்.

ஆங்கிலத்திலுள்ள மின்புத்தகங்கள்:

ஆங்கிலத்தில் லட்சக்கணக்கான மின்புத்தகங்கள் தற்போது கிடைக்கப் பெறுகின்றன. அவை PDF, EPUB, MOBI, AZW3. போன்ற வடிவங்களில் இருப்பதால், அவற்றை மேற்கூறிய கருவிகளைக் கொண்டு நாம் படித்துவிடலாம்.

தமிழிலுள்ள மின்புத்தகங்கள்:

தமிழில் சமீபத்திய புத்தகங்களெல்லாம் நமக்கு மின்புத்தகங்களாக கிடைக்கப்பெறுவதில்லை. ProjectMadurai.com எனும் குழு தமிழில் மின்புத்தகங்களை வெளியிடுவதற்கான ஓர் உன்னத சேவையில்

ஈடுபட்டுள்ளது. இந்தக் குழு இதுவரை வழங்கியுள்ள தமிழ் மின்புத்தகங்கள் அனைத்தும் PublicDomain-ல் உள்ளன. ஆனால் இவை மிகவும் பழைய புத்தகங்கள்.

சமீபத்திய புத்தகங்கள் ஏதும் இங்கு கிடைக்கப்பெறுவதில்லை.

சமீபத்திய புத்தகங்களை தமிழில் பெறுவது எப்படி?

அமேசான் கிண்டில் கருவியில் தமிழ் ஆதரவு தந்த பிறகு, தமிழ் மின்னூல்கள் அங்கே விற்பனைக்குக் கிடைக்கின்றன. ஆனால் அவற்றை நாம் பதிவிறக்க இயலாது. வேறு யாருக்கும் பகிர இயலாது.

சமீபகாலமாக பல்வேறு எழுத்தாளர்களும், பதிவர்களும், சமீபத்திய நிகழ்வுகளைப் பற்றிய விவரங்களைத் தமிழில் எழுதத் தொடங்கியுள்ளனர். அவை இலக்கியம், விளையாட்டு, கலாச்சாரம், உணவு, சினிமா, அரசியல், புகைப்படக்கலை, வணிகம் மற்றும் தகவல் தொழில்நுட்பம் போன்ற பல்வேறு தலைப்புகளின் கீழ் அமைகின்றன.

நாம் அவற்றையெல்லாம் ஒன்றாகச் சேர்த்து தமிழ் மின்புத்தகங்களை உருவாக்க உள்ளோம்.

அவ்வாறு உருவாக்கப்பட்ட மின்புத்தகங்கள் Creative Commons எனும் உரிமத்தின் கீழ் வெளியிடப்படும். இவ்வாறு வெளியிடுவதன் மூலம் அந்தப் புத்தகத்தை எழுதிய மூல ஆசிரியருக்கான உரிமைகள்

சட்டரீதியாகப் பாதுகாக்கப்படுகின்றன. அதே நேரத்தில் அந்த மின்புத்தகங்களை யார் வேண்டுமானாலும், யாருக்கு வேண்டுமானாலும், இலவசமாக வழங்கலாம்.

எனவே தமிழ் படிக்கும் வாசகர்கள் ஆயிரக்கணக்கில் சமீபத்திய தமிழ் மின்புத்தகங்களை இலவசமாகவே பெற்றுக் கொள்ள முடியும்.

தமிழிலிருக்கும் எந்த வலைப்பதிவிலிருந்து வேண்டுமானாலும் பதிவுகளை எடுக்கலாமா?

கூடாது.

ஒவ்வொரு வலைப்பதிவும் அதற்கென்றே ஒருசில அனுமதிகளைப் பெற்றிருக்கும். ஒரு வலைப்பதிவின் ஆசிரியர் அவரது பதிப்புகளை “யார் வேண்டுமானாலும் பயன்படுத்தலாம்” என்று குறிப்பிட்டிருந்தால் மட்டுமே அதனை நாம் பயன்படுத்த முடியும்.

அதாவது “Creative Commons” எனும் உரிமத்தின் கீழ் வரும் பதிப்புகளை மட்டுமே நாம் பயன்படுத்த முடியும்.

அப்படி இல்லாமல் “All Rights Reserved” எனும் உரிமத்தின் கீழ் இருக்கும் பதிப்புகளை நம்மால் பயன்படுத்த முடியாது.

வேண்டுமானால் “All Rights Reserved” என்று விளங்கும் வலைப்பதிவுகளைக் கொண்டிருக்கும் ஆசிரியருக்கு அவரது

பதிப்புகளை “Creative Commons” உரிமத்தின் கீழ் வெளியிடக்கோரி நாம் நமது வேண்டுகோளைத் தெரிவிக்கலாம். மேலும் அவரது படைப்புகள் அனைத்தும் அவருடைய பெயரின் கீழே தான் வெளியிடப்படும் எனும் உறுதியையும் நாம் அளிக்க வேண்டும்.

பொதுவாக புதுப்புது பதிவுகளை உருவாக்குவோருக்கு அவர்களது பதிவுகள் நிறைய வாசகர்களைச் சென்றடைய வேண்டும் என்ற எண்ணம் இருக்கும். நாம் அவர்களது படைப்புகளை எடுத்து இலவச மின்புத்தகங்களாக வழங்குவதற்கு நமக்கு
அவர்கள் அனுமதியளித்தால், உண்மையாகவே அவர்களது படைப்புகள் பெரும்பான்மையான மக்களைச் சென்றடையும். வாசகர்களுக்கும் நிறைய புத்தகங்கள் படிப்பதற்குக் கிடைக்கும்

வாசகர்கள் ஆசிரியர்களின் வலைப்பதிவு முகவரிகளில் கூட அவர்களுடைய படைப்புகளை தேடிக் கண்டுபிடித்து படிக்கலாம். ஆனால் நாங்கள் வாசகர்களின் சிரமத்தைக் குறைக்கும் வண்ணம் ஆசிரியர்களின் சிதறிய வலைப்பதிவுகளை ஒன்றாக இணைத்து ஒரு முழு மின்புத்தகங்களாக உருவாக்கும் வேலையைச் செய்கிறோம். மேலும் அவ்வாறு உருவாக்கப்பட்ட புத்தகங்களை “மின்புத்தகங்களைப் படிக்க உதவும் கருவிகள்”-க்கு ஏற்ற வண்ணம் வடிவமைக்கும் வேலையையும் செய்கிறோம்.

FREETAMILEBOOKS.COM

இந்த வலைத்தளத்தில்தான் பின்வரும் வடிவமைப்பில் மின்புத்தகங்கள் காணப்படும்.

PDF for desktop, PDF for 6” devices, EPUB, AZW3, ODT

இந்த வலைதளத்திலிருந்து யார் வேண்டுமானாலும் மின்புத்தகங்களை இலவசமாகப் பதிவிறக்கம்(download) செய்து கொள்ளலாம்.

அவ்வாறு பதிவிறக்கம்(download) செய்யப்பட்ட புத்தகங்களை யாருக்கு வேண்டுமானாலும் இலவசமாக வழங்கலாம்.

இதில் நீங்கள் பங்களிக்க விரும்புகிறீர்களா?

நீங்கள் செய்யவேண்டியதெல்லாம் தமிழில் எழுதப்பட்டிருக்கும் வலைப்பதிவுகளிலிருந்து பதிவுகளை
எடுத்து, அவற்றை LibreOffice/MS Office போன்ற wordprocessor-ல் போட்டு ஓர் எளிய மின்புத்தகமாக மாற்றி எங்களுக்கு அனுப்பவும்.

அவ்வளவுதான்!

மேலும் சில பங்களிப்புகள் பின்வருமாறு:

1. ஒருசில பதிவர்கள்/எழுத்தாளர்களுக்கு அவர்களது படைப்புகளை “Creative Commons” உரிமத்தின்கீழ் வெளியிடக்கோரி மின்னஞ்சல் அனுப்புதல்

2. தன்னார்வலர்களால் அனுப்பப்பட்ட மின்புத்தகங்களின் உரிமைகளையும் தரத்தையும் பரிசோதித்தல்

3. சோதனைகள் முடிந்து அனுமதி வழங்கப்பட்ட தரமான மின்புத்தகங்களை நமது வலைதளத்தில் பதிவேற்றம் செய்தல்

விருப்பமுள்ளவர்கள் freetamilebooksteam@gmail.com எனும் முகவரிக்கு மின்னஞ்சல் அனுப்பவும்.

இந்தத் திட்டத்தின் மூலம் பணம் சம்பாதிப்பவர்கள் யார்?

யாருமில்லை.

இந்த வலைத்தளம் முழுக்க முழுக்க தன்னார்வலர்களால் செயல்படுகின்ற ஒரு வலைத்தளம் ஆகும். இதன் ஒரே நோக்கம் என்னவெனில் தமிழில் நிறைய மின்புத்தகங்களை உருவாக்குவதும், அவற்றை இலவசமாக பயனர்களுக்கு வழங்குவதுமே ஆகும்.

மேலும் இவ்வாறு உருவாக்கப்பட்ட மின்புத்தகங்கள், ebook reader ஏற்றுக்கொள்ளும் வடிவமைப்பில் அமையும்.

இத்திட்டத்தால் பதிப்புகளை எழுதிக்கொடுக்கும் ஆசிரியர்/பதிவருக்கு என்ன லாபம்?

ஆசிரியர்/பதிவர்கள் இத்திட்டத்தின் மூலம் எந்தவிதமான தொகையும் பெறப்போவதில்லை. ஏனெனில், அவர்கள் புதிதாக இதற்கென்று எந்தஒரு பதிவையும் எழுதித்தரப்போவதில்லை.

ஏற்கனவே அவர்கள் எழுதி வெளியிட்டிருக்கும் பதிவுகளை எடுத்துத்தான் நாம் மின்புத்தகமாக வெளியிடப்போகிறோம்.

அதாவது அவரவர்களின் வலைதளத்தில் இந்தப் பதிவுகள் அனைத்தும் இலவசமாகவே கிடைக்கப்பெற்றாலும், அவற்றையெல்லாம் ஒன்றாகத் தொகுத்து ebook reader போன்ற கருவிகளில் படிக்கும் விதத்தில் மாற்றித் தரும் வேலையை இந்தத் திட்டம் செய்கிறது.

தற்போது மக்கள் பெரிய அளவில் tablets மற்றும் ebook readers போன்ற கருவிகளை நாடிச் செல்வதால் அவர்களை நெருங்குவதற்கு இது ஒரு நல்ல வாய்ப்பாக அமையும்.

நகல் எடுப்பதை அனுமதிக்கும் வலைதளங்கள் ஏதேனும் தமிழில் உள்ளதா?

உள்ளது.

பின்வரும் தமிழில் உள்ள வலைதளங்கள் நகல் எடுப்பதினை அனுமதிக்கின்றன.

1. http://www.vinavu.com
2. http://www.badriseshadri.in
3. http://maattru.com
4. http://www.kaniyam.com
5. http://blog.ravidreams.net

எவ்வாறு ஓர் எழுத்தாளரிடம் CREATIVE COMMONS உரிமத்தின் கீழ் அவரது படைப்புகளை வெளியிடுமாறு கூறுவது?

இதற்கு பின்வருமாறு ஒரு மின்னஞ்சலை அனுப்ப வேண்டும்.

துவக்கம்

உங்களது வலைத்தளம் அருமை (வலைதளத்தின் பெயர்).

தற்போது படிப்பதற்கு உபயோகப்படும் கருவிகளாக Mobiles மற்றும் பல்வேறு கையிருப்புக் கருவிகளின் எண்ணிக்கை அதிகரித்து வந்துள்ளது.

இந்நிலையில் நாங்கள் http://www.FreeTamilEbooks.com எனும் வலைதளத்தில், பல்வேறு தமிழ் மின்புத்தகங்களை வெவ்வேறு துறைகளின் கீழ் சேகரிப்பதற்கான ஒரு புதிய திட்டத்தில் ஈடுபட்டுள்ளோம்.

இங்கு சேகரிக்கப்படும் மின்புத்தகங்கள் பல்வேறு கணிணிக் கருவிகளான Desktop,ebook readers like kindl, nook, mobiles, tablets with android, iOS போன்றவற்றில் படிக்கும் வண்ணம் அமையும். அதாவது இத்தகைய கருவிகள் support செய்யும் odt, pdf, ebub, azw போன்ற வடிவமைப்பில் புத்தகங்கள் அமையும்.

இதற்காக நாங்கள் உங்களது வலைதளத்திலிருந்து பதிவுகளை பெற விரும்புகிறோம். இதன் மூலம் உங்களது பதிவுகள் உலகளவில் இருக்கும் வாசகர்களின் கருவிகளை நேரடியாகச் சென்றடையும்.

எனவே உங்களது வலைதளத்திலிருந்து பதிவுகளை பிரதியெடுப்பதற்கும் அவற்றை மின்புத்தகங்களாக மாற்றுவதற்கும் உங்களது அனுமதியை வேண்டுகிறோம்.

இவ்வாறு உருவாக்கப்பட்ட மின்புத்தகங்களில் கண்டிப்பாக ஆசிரியராக உங்களின் பெயரும் மற்றும் உங்களது வலைதள முகவரியும் இடம்பெறும். மேலும் இவை “Creative Commons” உரிமத்தின் கீழ் மட்டும்தான் வெளியிடப்படும் எனும் உறுதியையும் அளிக்கிறோம்.

http://creativecommons.org/licenses/

நீங்கள் எங்களை பின்வரும் முகவரிகளில் தொடர்பு கொள்ளலாம்.

e-mail : FREETAMILEBOOKSTEAM@GMAIL.COM
FB : https://www.facebook.com/FreeTamilEbooks

G plus: https://plus.google.com/communities/108817760492177970948

நன்றி.

முடிவு

மேற்கூறியவாறு ஒரு மின்னஞ்சலை உங்களுக்குத் தெரிந்த அனைத்து எழுத்தாளர்களுக்கும் அனுப்பி அவர்களிடமிருந்து அனுமதியைப் பெறுங்கள்.

முடிந்தால் அவர்களையும் “Creative Commons License”-ஐ அவர்களுடைய வலைதளத்தில் பயன்படுத்தச் சொல்லுங்கள்.

கடைசியாக அவர்கள் உங்களுக்கு அனுமதி அளித்து அனுப்பியிருக்கும் மின்னஞ்சலைFREETAMILEBOOKSTEAM@GMAIL.COM எனும் முகவரிக்கு அனுப்பி வையுங்கள்.

ஓர் எழுத்தாளர் உங்களது உங்களது வேண்டுகோளை மறுக்கும் பட்சத்தில் என்ன செய்வது?

அவர்களையும் அவர்களது படைப்புகளையும் அப்படியே விட்டுவிட வேண்டும்.

ஒருசிலருக்கு அவர்களுடைய சொந்த முயற்சியில் மின்புத்தகம் தயாரிக்கும் எண்ணம்கூட இருக்கும். ஆகவே அவர்களை நாம் மீண்டும் மீண்டும் தொந்தரவு செய்யக் கூடாது.

அவர்களை அப்படியே விட்டுவிட்டு அடுத்தடுத்த எழுத்தாளர்களை நோக்கி நமது முயற்சியைத் தொடர வேண்டும்.

மின்புத்தகங்கள் எவ்வாறு அமைய வேண்டும்?

ஒவ்வொருவரது வலைத்தளத்திலும் குறைந்தபட்சம் நூற்றுக்கணக்கில் பதிவுகள் காணப்படும். அவை வகைப்படுத்தப்பட்டோ அல்லது வகைப்படுத்தப் படாமலோ இருக்கும்.

நாம் அவற்றையெல்லாம் ஒன்றாகத் திரட்டி ஒரு பொதுவான தலைப்பின்கீழ் வகைப்படுத்தி மின்புத்தகங்களாகத் தயாரிக்கலாம். அவ்வாறு வகைப்படுத்தப்படும் மின்புத்தகங்களை பகுதி-I பகுதி-II என்றும் கூட தனித்தனியே பிரித்துக் கொடுக்கலாம்.

தவிர்க்க வேண்டியவைகள் யாவை?

இனம், பாலியல் மற்றும் வன்முறை போன்றவற்றைத் தூண்டும் வகையான பதிவுகள் தவிர்க்கப்பட வேண்டும்.

எங்களைத் தொடர்பு கொள்வது எப்படி?

நீங்கள் பின்வரும் முகவரிகளில் எங்களைத் தொடர்பு கொள்ளலாம்.

- EMAIL : FREETAMILEBOOKSTEAM@GMAIL.COM
- Facebook: https://www.facebook.com/FreeTamilEbooks
- Google Plus: https://plus.google.com/communities/108817760492177970948

இத்திட்டத்தில் ஈடுபட்டுள்ளவர்கள் யார்?

குழு – http://freetamilebooks.com/meet-the-team/

SUPPORTED BY

கணியம் அறக்கட்டளை http://kaniyam.com/foundation

கணியம் அறக்கட்டளை



தொலை நோக்கு – Vision

தமிழ் மொழி மற்றும் இனக்குழுக்கள் சார்ந்த மெய்நிகர்வளங்கள், கருவிகள் மற்றும் அறிவுத்தொகுதிகள், அனைவருக்கும் கட்டற்ற அணுக்கத்தில் கிடைக்கும் சூழல்

பணி இலக்கு – Mission

அறிவியல் மற்றும் சமூகப் பொருளாதார வளர்ச்சிக்கு ஒப்ப, தமிழ் மொழியின் பயன்பாடு வளர்வதை உறுதிப்படுத்துவதும், அனைத்து அறிவுத் தொகுதிகளும், வளங்களும் கட்டற்ற அணுக்கத்தில் அனைவருக்கும் கிடைக்கச்செய்தலும்.

தற்போதைய செயல்கள்

- கணியம் மின்னிதழ் – http://kaniyam.com
- கிரியேட்டிவ் காமன்சு உரிமையில் இலவச தமிழ் மின்னூல்கள் – http://FreeTamilEbooks.com

கட்டற்ற மென்பொருட்கள்

- உரை ஒலி மாற்றி – Text to Speech
- எழுத்துணரி – Optical Character Recognition
- விக்கிமூலத்துக்கான எழுத்துணரி
- மின்னூல்கள் கிண்டில் கருவிக்கு அனுப்புதல் – Send2Kindle
- விக்கிப்பீடியாவிற்கான சிறு கருவிகள்
- மின்னூல்கள் உருவாக்கும் கருவி
- உரை ஒலி மாற்றி – இணைய செயலி
- சங்க இலக்கியம் – ஆன்டிராய்டு செயலி
- FreeTamilEbooks – ஆன்டிராய்டு செயலி

- FreeTamilEbooks – ஐஓஎஸ் செயலி
- WikisourceEbooksReportஇந்திய மொழிகளுக்ககான விக்கிமூலம் மின்னூல்கள் பதிவிறக்கப் பட்டியல்
- FreeTamilEbooks.com – Download counter மின்னூல்கள் பதிவிறக்கப் பட்டியல்

அடுத்த திட்டங்கள்/மென்பொருட்கள்

- விக்கி மூலத்தில் உள்ள மின்னூல்களை பகுதிநேர/முழு நேரப் பணியாளர்கள் மூலம் விரைந்து பிழை திருத்துதல்
- முழு நேர நிரலரை பணியமர்த்தி பல்வேறு கட்டற்ற மென்பொருட்கள் உருவாக்குதல்
- தமிழ் NLP க்கான பயிற்சிப் பட்டறைகள் நடத்துதல்
- கணியம் வாசகர் வட்டம் உருவாக்குதல்
- கட்டற்ற மென்பொருட்கள், கிரியேட்டிவ் காமன்சு உரிமையில் வளங்களை உருவாக்குபவர்களைக் கண்டறிந்து ஊக்குவித்தல்

- கணியம் இதழில் அதிக பங்களிப்பாளர்களை உருவாக்குதல், பயிற்சி அளித்தல்
- மின்னூலாக்கத்துக்கு ஒரு இணையதள செயலி
- எழுத்துணரிக்கு ஒரு இணையதள செயலி
- தமிழ் ஒலியோடைகள் உருவாக்கி வெளியிடுதல்
- http://OpenStreetMap.org ல் உள்ள இடம், தெரு, ஊர் பெயர்களை தமிழாக்கம் செய்தல்
- தமிழ்நாடு முழுவதையும் http://OpenStreetMap.org ல் வரைதல்
- குழந்தைக் கதைகளை ஒலி வடிவில் வழங்குதல்
- http://Ta.wiktionary.org ஐ ஒழுங்குபடுத்தி API க்கு தோதாக மாற்றுதல்
- http://Ta.wiktionary.org க்காக ஒலிப்பதிவு செய்யும் செயலி உருவாக்குதல்
- தமிழ் எழுத்துப் பிழைத்திருத்தி உருவாக்குதல்
- தமிழ் வேர்ச்சொல் காணும் கருவி உருவாக்குதல்

- எல்லா http://FreeTamilEbooks.com மின்னூல்களையும் Google Play Books, GoodReads.com ல் ஏற்றுதல்
- தமிழ் தட்டச்சு கற்க இணைய செயலி உருவாக்குதல்
- தமிழ் எழுதவும் படிக்கவும் கற்ற இணைய செயலி உருவாக்குதல் (aamozish.com/Course_preface போல)

மேற்கண்ட திட்டங்கள், மென்பொருட்களை உருவாக்கி செயல்படுத்த உங்கள் அனைவரின் ஆதரவும் தேவை. உங்களால் எவ்வாறேனும் பங்களிக்க இயலும் எனில் உங்கள் விவரங்களை kaniyamfoundation@gmail.com க்கு மின்னஞ்சல் அனுப்புங்கள்.

வெளிப்படைத்தன்மை

கணியம் அறக்கட்டளையின் செயல்கள், திட்டங்கள், மென்பொருட்கள் யாவும் அனைவருக்கும் பொதுவானதாகவும், 100% வெளிப்படைத்தன்மையுடனும் இருக்கும்.இந்த இணைப்பில் செயல்களையும், இந்த இணைப்பில் மாத அறிக்கை, வரவு செலவு விவரங்களுடனும் காணலாம்.

கணியம் அறக்கட்டளையில் உருவாக்கப்படும் மென்பொருட்கள் யாவும் கட்டற்ற மென்பொருட்களாக மூல நிரலுடன், GNU GPL, Apache, BSD, MIT, Mozilla ஆகிய உரிமைகளில் ஒன்றாக வெளியிடப்படும். உருவாக்கப்படும் பிற வளங்கள், புகைப்படங்கள், ஒலிக்கோப்புகள், காணொளிகள், மின்னூல்கள், கட்டுரைகள் யாவும் யாவரும் பகிரும், பயன்படுத்தும் வகையில் கிரியேட்டிவ் காமன்சு உரிமையில் இருக்கும்.

நன்கொடை

உங்கள் நன்கொடைகள் தமிழுக்கான கட்டற்ற வளங்களை உருவாக்கும் செயல்களை சிறந்த வகையில் விரைந்து செய்ய ஊக்குவிக்கும்.

பின்வரும் வங்கிக் கணக்கில் உங்கள் நன்கொடைகளை அனுப்பி, உடனே விவரங்களை kaniyamfoundation@gmail.com க்கு மின்னஞ்சல் அனுப்புங்கள்.

Kaniyam Foundation
Account Number : 606 1010 100 502 79
Union Bank Of India
West Tambaram, Chennai
IFSC – UBIN0560618
Account Type : Current Account

UPI செயலிகளுக்கான QR Code



BHIM UPI Payments Accepted at

Kaniyam Foundation



Account Number : 606101010050279, IFSC Code: UBIN0560618

Scan and Pay using any UPI supported Apps

குறிப்பு: சில UPI செயலிகளில் இந்த QR Code வேலை செய்யாமல் போகலாம். அச்சமயம் மேலே உள்ள வங்கிக் கணக்கு எண், IFSC code ஐ பயன்படுத்தவும்.

Note: Sometimes UPI does not work properly, in that case kindly use Account number and IFSC code for internet banking.